Sorting and Searching:

A Cookbook

Thomas Niemann

+--+

***** Contents *****

Preface

This booklet contains a collection of sorting and searching algorithms. While many books on data structures describe sorting and searching algorithms, most assume a background in calculus and probability theory. Although a formal presentation and proof of asymptotic behavior is important, a more intuitive explanation is often possible.

The way each algorithm works is described in easy-to-understand terms. It is assumed that you have the knowledge equivalent to an introductory course in C or Pascal. In particular, you should be familiar with arrays and have a basic understanding of pointers. The material is presented in an orderly fashion, beginning with easy concepts and progressing to more complex ideas. Even though this collection is intended for beginners, more advanced users may benefit from some of the insights offered. In particular, the sections on hash tables and skip
lists should prove interesting.

Santa Cruz, California
Thomas Niemann
March, 1995

1. Introduction

Arrays and linked lists are two basic data structures used to store information. We may wish to search, insert or delete records in a database based on a key value. This section examines the performance of these operations on arrays and linked lists.

Arrays

Figure 1.1 shows an array, seven elements long, containing numeric values. To search the array sequentially, we may use the algorithm in Figure 1.2. The maximum number of comparisons is 7, and occurs when the key we are searching for is in A[6]. If the data is sorted, a binary search may be done (Figure 1.3). Variables Lb and Ub keep track of the lower bound and upper bound of the array, respectively. We begin by examining the middle element of the array. If the key we are searching for is less than the middle element, then it must reside in the top half of the array. Thus, we set Ub to (M Ñ 1). This restricts our next iteration through the loop to the top half of the array. In this way, each iteration halves the size of the array to be searched. For example, the first iteration will leave 3 items to test. After the second iteration, there will be 1 item left to test. Thus it takes only three iterations to find any number.

Figure 1.1: An Array

This is a powerful method. For example, if the array size is 1023, we can narrow the search to 511 items in one comparison. Another comparison, and we're looking at only 255 elements. In fact, only 10 comparisons are needed to search an array containing 1023 elements.

In addition to searching, we may wish to insert or delete entries. Unfortunately, an array is not a good arrangement for these operations. For example, to insert the number 18 in Figure 1.1, we would need to shift A[3]ÉA[6] down by one slot. Then we could copy number 18 into A[3]. A similar problem arises when deleting numbers. To improve the efficiency of insert and delete operations, linked lists may be used.

```
int function SequentialSearch (Array A , int Lb , int Ub , int Key );
      begin
            for i = Lb to Ub do
            if A ( i ) =  Key then
                  return i ;
            return Ñ1;
      end;
```

Figure 1.2: Sequential Search

```
int function BinarySearch (Array A , int Lb , int Ub , int Key );
      begin
      do forever
            M = ( Lb + Ub )/2;
            if ( Key < A[M]) then
```

```
                        Ub = M Ñ 1;
                  else if (Key > A[M]) then
                        Lb = M + 1;
                  else
                        return M ;
                  if (Lb > Ub ) then
                        return Ñ1;
            end;
```

Figure 1.3: Binary Search

Linked Lists

In Figure 1.4 we have the same values stored in a linked list. Assuming pointers X and P, as shown in the figure, value 18 may be inserted as follows:

```
X->Next = P->Next;
P->Next = X;
```

Insertion (and deletion) are very efficient using linked lists. You may be wondering how P was set in the first place. Well, we had to search the list in a sequential fashion to find the insertion point for X. Thus, while we improved our insert and delete performance, it has been at the expense of search time.

Figure 1.4: A Linked List

Timing Estimates

Several methods may be used to compare the performance of algorithms. One way is simply to run several tests for each algorithm and compare the timings. Another way is to estimate the time required. For example, we may state that search time is O(n) (big-ohÊofÊn). This means that, for large n, search time is no greater than the number of items n in the list. The big-O notation does not describe the exact time that an algorithm takes, but only indicates an upper bound on execution time within a constant factor. If an algorithm takes O(n2) time, then execution time grows no worse than the square of the size of the list. To see the effect this has, Table 1.1 illustrates growth rates for various functions. A growth rate of O(lg n) occurs for algorithms similar to the binary search. The lg (logarithm, base 2) function increases by one when n is doubled. Recall that we can search twice as many items with one more comparison in the binary search. Thus the binary search is a O(lg n) algorithm.

1.1: Growth Rates

If the values in Table 1.1 represented microseconds, then a O(lg n) algorithm may take 20 microseconds to process 1,048,476 items, a O(n1.25) algorithm might take 33 seconds, and a O(n2) algorithm might take up to 12 days! In the following chapters a timing estimate for each algorithm, using big-O notation, will be included. For a more formal derivation of these formulas you may wish to consult the references.

Summary

As we have seen, sorted arrays may be searched efficiently using a binary search. However, we must have a sorted array to start with. In the next section various ways to sort arrays will be examined. It turns out that this is computationally expensive, and considerable research has been done to make sorting algorithms as efficient as possible.

Linked lists improved the efficiency of insert and delete operations, but searches were sequential and time-consuming. Algorithms exist that do all three operations efficiently, and they will be the discussed in the section on dictionaries.

1. Sorting

a) Insertion Sort

One of the simplest methods to sort an array is sort by insertion. An example of an insertion sort occurs in everyday life while playing cards. To sort the cards in your hand you extract a card, shift the remaining cards, and then insert the extracted card in the correct place. This process is repeated until all the cards are in the correct sequence. Both average and worst-case time is O(n2). For further reading, consult Knuth[1].

Theory

In Figure 2.1(a) we extract the 3. Then the above elements are shifted down until we find the correct place to insert the 3. This process repeats in Figure 2.1(b) for the number 1. Finally, in Figure 2.1(c), we complete the sort by inserting 2 in the correct place.

Assuming there are n elements in the array, we must index through n Ñ 1 entries. For each entry, we may need to examine and shift up to n Ñ 1 other entries. For this reason, sorting is a time-consuming process.

The insertion sort is an in-place sort. That is, we sort the array in-place.
No extra memory is required. The insertion sort is also a stable sort. Stable sorts retain the original ordering of keys when identical keys are present in the input data.

Figure 2.1: Insertion Sort

Implementation

An ANSI-C implementation for insertion sort may be found in Section 4.1 (page). Typedef T and comparison operator CompGT should be altered to reflect the data stored in the table. Pointer arithmetic was used, rather than array references, for efficiency.

a) Shell Sort

Shell sort, developed by Donald L. Shell, is a non-stable in-place sort. Shell sort improves on the efficiency of insertion sort by quickly shifting values to their destination. Average sort time is O(n1.25), while worst-case time is O(n1.5). For further reading, consult Knuth[1].

Theory

In Figure 2.2(a) we have an example of sorting by insertion. First we extract 1, shift 3 and 5 down one slot, and then insert the 1. Thus, two shifts were required. In the next frame, two shifts are required before we can insert the 2. The process continues until the last frame, where a total of 2 + 2 + 1 = 5 shifts have been made.

In Figure 2.2(b) an example of shell sort is illustrated. We begin by doing an insertion sort using a spacing of two. In the first frame we examine numbers 3-1. Extracting 1, we shift 3 down one slot for a shift count of 1. Next we examine numbers 5-2. We extract 2, shift 5 down, and then insert 2. After sorting with a spacing of two, a final pass is made with a spacing of one. This is simply the traditional insertion sort. The total shift count using shell sort is 1+1+1 = 3. By using an initial spacing larger than one, we were able to quickly shift values to their proper destination.

Figure 2.2: Shell Sort

To implement shell sort, various spacings may be used. Typically the array is sorted with a large spacing, the spacing reduced, and the array sorted again. On the final sort, spacing is one. Although shell sort is easy to comprehend, formal analysis is difficult. In particular, optimal spacing values elude theoreticians. Knuth[1] has experimented with several values and recommends that spacing (h) for an array of size N be based on the following formula:

Thus, values of h are computed as follows:

To sort 100 items we first find an hs such that hs ³ 100. For 100 items, h5 is selected. Our final value (ht) is two steps lower, or h3. Thus, our sequence of h values will be 13-4-1. Once the initial h value has been determined, subsequent values may be calculated using the formula

Implementation

An ANSI-C implementation of shell sort may be found in Section 4.2 (page).

Typedef T and comparison operator CompGT should be altered to reflect the data stored in the array. When computing h, care must be taken to avoid underflows or overflows. The central portion of the algorithm is an insertion sort with a spacing of h. To terminate the inner loop correctly, it is necessary to compare J before decrement. Otherwise, pointer values may wrap through zero, resulting in unexpected behavior.

b) Quicksort

While the shell sort algorithm is significantly better than insertion sort, there is still room for improvement. One of the most popular sorting algorithms is quicksort. Quicksort executes in O(n lg n) on average, and O(n2) in the worst-case. However, with proper precautions, worst-case behavior is very unlikely. Quicksort is a non-stable sort. It is not an in-place sort as stack space is required. For further reading, consult Cormen[2].

Theory

The quicksort algorithm works by partitioning the array to be sorted, then recursively sorting each partition. In Partition (Figure 2.3), one of the array elements is selected as a pivot value. Values smaller than the pivot value are placed to the left of the pivot, while larger values are placed to the right.

```
int function Partition (Array A, int Lb, int Ub);
      begin
            select a pivot from A[Lb]ÉA[Ub];
            reorder A[Lb]ÉA[Ub] such that:
                  all values to the left of the pivot are £ pivot
                  all values to the right of the pivot are ³ pivot
            return pivot position;
      end;

procedure QuickSort (Array A, int Lb, int Ub);
      begin
            if Lb Ub then
                  M = Partition (A, Lb, Ub);
                  QuickSort (A, Lb, M Ñ 1);
                  QuickSort (A, M + 1, Ub);
      end;
```

Figure 2.3: Quicksort Algorithm

In Figure 2.4(a), the pivot selected is 3. Indices are run starting at both ends of the array. Index i starts on the left and selects an element that is larger than the pivot, while index j starts on the right and selects an element that is smaller than the pivot. These elements are then exchanged, as is shown in Figure 2.4(b). QuickSort recursively sorts the two subarrays, resulting in the array shown in Figure 2.4(c).

Figure 2.4: Quicksort Example

As the process proceeds, it may be necessary to move the pivot so that correct ordering is maintained. In this manner, QuickSort succeeds in sorting the array.

If we're lucky the pivot selected will be the median of all values, thus equally

dividing the array. For a moment, let's assume that this is the case. Since the array is split in half at each step, and Partition must eventually examine all n elements, the run time is O(n lg n).

To find a pivot value, Partition could simply select the first element (A[Lb]). All other values would be compared to the pivot value, and placed either to the left or right of the pivot as appropriate. However, there is one case that fails miserably. Suppose the array was originally in order. Partition would always select the lowest value as a pivot and split the array with one element in the left partition, and Ub Ñ Lb elements in the other. Each recursive call to quicksort would only diminish the size of the array to be sorted by one. Thus, n recursive calls would be required to do the sort, resulting in a O(n2) run time. One solution to this problem is to randomly select an item as a pivot. This would make it extremely unlikely that worst-case behavior would occur.

Implementation

An ANSI-C implementation of the quicksort algorithm may be found in Section 4.3 (pageÊ). Typedef T and comparison operator CompGT should be altered to reflect the data stored in the array. Several enhancements have been made to the basic quicksort algorithm:

- The center element is selected as a pivot in Partition. If the list is partially ordered, this will be a good choice. Worst-case behavior occurs when the center element happens to be the largest or smallest element each time Partition is invoked.

- For short arrays, InsertSort is called. Due to recursion and other overhead, quicksort is not an efficient algorithm to use on small arrays. Thus, any array with fewer than 12 elements is sorted using an insertion sort. The optimal cutoff value is not critical and varies based on the quality of generated code.

- Tail recursion occurs when the last statement in a function is a call to the function itself. Tail recursion may be replaced by iteration which results in a better utilization of stack space. This has been done with

the second call to QuickSort in Figure 2.3.

· After an array is partitioned, the smallest partition is sorted first.
This results in a better utilization of stack space, as short partitions
are quickly sorted and dispensed with.

· Pointer arithmetic, rather than array indices, is used for efficient execution.

Also included is a listing for qsort (Section 4.4, page), an ANSI-C standard
library function usually implemented with quicksort. For this implementation,
recursive calls were replaced by explicit stack operations. Table 2.1 shows
timing statistics and stack utilization before and after the enhancements
were applied.

b) Comparison of Methods

In this section we will compare the sorting algorithms covered: insertion sort,
shell sort and quicksort. There are several factors that influence the choice
of a sorting algorithm:

· Stable sort. Recall that a stable sort will leave identical keys in the
same relative position in the sorted output. Insertion sort is the only
algorithm covered that is stable.

· Space. An in-place sort does not require any extra space to accomplish its
task. Both insertion sort and shell sort are in-place sorts. Quicksort
requires stack space for recursion, and thus is not an in-place sort.
However, the amount required was considerably reduced by tinkering with the
algorithm.

· Time. The time required to sort a dataset can easily become astronomical
(Table 1.1). Table 2.2 shows the relative timings for each method. Actual
timing tests are described below.

· Simplicity. The number of statements required for each algorithm may be
found in Table 2.2. Simpler algorithms result in fewer programming errors.

The time required to sort a randomly ordered dataset is shown in Table 2.3.

1. Dictionaries

a) Hash Tables

A dictionary requires that search, insert and delete operations be supported. One of the most effective ways to implement a dictionary is through the use of hash tables. Average time to search for an element is O(1), while worst-case time is O(n). Cormen[2] and Knuth[1] both contain excellent discussions on hashing. In case you decide to read more material on this topic, you may want to know some terminology. The technique presented here is chaining, also known as open hashing[3]. An alternative technique, known as closed hashing[3], or open addressing[1], is not presented. Got that?

Theory

A hash table is simply an array that is addressed via a hash function. For example, in Figure 3.1, HashTable is an array with 8 elements. Each element is a pointer to a linked list of numeric data. The hash function for this example simply divides the data key by 8, and uses the remainder as an index into the table. This yields a number from 0 to 7. Since the range of indices for HashTable is 0 to 7, we are guaranteed that the index is valid.

Figure 3.1: A Hash Table

To insert a new item in the table, we hash the key to determine which list the item goes on, and then insert the item at the beginning of the list. For example, to insert 11, we divide 11 by 8 giving a remainder of 3. Thus, 11 goes on the list starting at HashTable[3]. To find a number, we hash the number and chain down the correct list to see if it is in the table. To delete a number, we find the number and remove the node from the linked list.

If the hash function is uniform, or equally distributes the data keys among the hash table indices, then hashing effectively subdivides the list to be searched. Worst-case behavior occurs when all keys hash to the same index. Then we simply have a single linked list that must be sequentially scanned. Consequently, it is important to choose a good hash function. Several methods may be used to hash key values. To illustrate the techniques, I will assume unsigned char is 8-bits, unsigned short int is 16-bits and unsigned long int is 32-bits.

· Division method (tablesize = prime). This technique was used in the preceding example. A HashValue, from 0 to (HashTableSize - 1), is computed by dividing the key value by the size of the hash table and taking the remainder. For example:

```
typedef int HashIndexType;

HashIndexType Hash(int Key) {
    return Key % HashTableSize;
}
```

· Selecting an appropriate HashTableSize is important to the success of this method. For example, a HashTableSize of two would yield even hash values for even Keys, and odd hash values for odd Keys. This is an undesirable property, as all keys would hash to the same value if they happened to be even. If HashTableSize is a power of two, then the hash function simply selects a subset of the Key bits as the table index. To obtain a more random scattering, HashTableSize should be a prime number not too close to a power of two.

· Multiplication method (tablesize = 2n). The multiplication method may be used for a HashTableSize that is a power of 2. The Key is multiplied by a constant, and then the necessary bits are extracted to index into the table. Knuth[1] recommends using the golden ratio, or , as the constant. The following definitions may be used for the multiplication method:

```
/* 8-bit index */
typedef unsigned char HashIndexType;
static const HashIndexType K = 158;

/* 16-bit index */
typedef unsigned short int HashIndexType;
static const HashIndexType K = 40503;
```

```
/* 32-bit index */
typedef unsigned long int HashIndexType;
static const HashIndexType K = 2654435769;

/* w=bitwidth(HashIndexType), size of table=2**m */
static const int S = w - m;
HashIndexType HashValue = (HashIndexType)(K * Key) >> S;
```

For example, if HashTableSize is 1024 (210), then a 16-bit index is sufficient and S would be assigned a value of 16 Ñ 10 = 6. Thus, we have:

```
typedef unsigned short int HashIndexType;

HashIndexType Hash(int Key) {
    static const HashIndexType K = 40503;
    static const int S = 6;
    return (HashIndexType)(K * Key) >> S;
}
```

· Variable string addition method (tablesize = 256). To hash a variable-length string, each character is added, modulo 256, to a total. A HashValue, range 0-255, is computed.

```
typedef unsigned char HashIndexType;

HashIndexType Hash(char *str) {
    HashIndexType h = 0;
    while (*str) h += *str++;
    return h;
}
```

· Variable string exclusive-or method (tablesize = 256). This method is similar to the addition method, but successfully distinguishes similar words and anagrams. To obtain a hash value in the range 0-255, all bytes in the string are exclusive-or'd together. However, in the process of doing each exclusive-or, a random component is introduced.

```
typedef unsigned char HashIndexType;
unsigned char Rand8[256];

HashIndexType Hash(char *str) {
    unsigned char h = 0;
    while (*str) h = Rand8[h ^ *str++];
    return h;
}
```

Rand8 is a table of 256 8-bit unique random numbers. The exact ordering is not critical. The exclusive-or method has its basis in cryptography, and is quite effective[4].

· Variable string exclusive-or method (tablesize £ 65536). If we hash the string twice, we may derive a hash value for an arbitrary table size up to 65536. The second time the string is hashed, one is added to the first character. Then the two 8-bit hash values are concatenated together to form a 16-bit hash value.

```
typedef unsigned short int HashIndexType;
unsigned char Rand8[256];

HashIndexType Hash(char *str) {
    HashIndexType h;
    unsigned char h1, h2;

    if (*str == 0) return 0;
    h1 = *str; h2 = *str + 1;
    str++;
    while (*str) {
        h1 = Rand8[h1 ^ *str];
        h2 = Rand8[h2 ^ *str];
```

```
        str++;
    }

    /* h is in range 0..65535 */
    h = ((HashIndexType)h1 << 8)|(HashIndexType)h2;
    /* use division method to scale */
    return h % HashTableSize
}
```

Assuming n data items, the hash table size should be large enough to accommodate a reasonable number of entries. As seen in Table 3.1, a small table size substantially increases the average time to find a key. A hash table may be viewed as a collection of linked lists. As the table becomes larger, the number of lists increases, and the average number of nodes on each list decreases. If the table size is 1, then the table is really a single linked list of length n. Assuming a perfect hash function, a table size of 2 has two lists of length n/2. If the table size is 100, then we have 100 lists of length n/100. This considerably reduces the length of the list to be searched. As we see in Table 3.1, there is much leeway in the choice of table size.

size time size time 1 869 128 9 2 432 256 6 4 214 512 4 8 106 1024 4 16 54 2048 3 32 28 4096 3 64 15 8192 3 Table 3.1: HashTableSize vs. Average Search Time (ms), 4096 entries

Implementation

An ANSI-C implementation of a hash table may be found in Section 4.5 (page). Typedef T and comparison operator CompEQ should be altered to reflect the data stored in the table. HashTableSize must be determined and the HashTable allocated. The division method was used in the Hash function. InsertNode allocates a new node and inserts it in the table. DeleteNode deletes and frees a node from the table. FindNode searches the table for a particular value.

a) Binary Search Trees

In Section 1 we used the binary search algorithm to find data stored in an array. This method is very effective, as each iteration reduced the number of items to search by one- half. However, since data was stored in an array, insertions and deletions were not efficient. Binary search trees store data in nodes that are linked in a tree-like fashion. For randomly inserted data, search time is O(lg n). Worst-case behavior occurs when ordered data is inserted. In this case the search time is O(n). See Cormen[2] for a more detailed description.

Theory

A binary search tree is a tree where each node has a left and right child. Either child, or both children, may be missing. Figure 3.2 illustrates a binary search tree. Assuming Key represents the value of a given node, then a binary search tree also has the following property: all children to the left of the node have values smaller than Key, and all children to the right of the node have values larger than Key. The top of a tree is known as the root, and the exposed nodes at the bottom are known as leaves. In Figure 3.2, the root is node 20 and the leaves are nodes 4, 16, 37 and 43. The height of a tree is the length of the longest path from root to leaf. For this example the tree height is 2.

Figure 3.2: A Binary Search Tree

To search a tree for a given value, we start at the root and work down. For example, to search for 16, we first note that 16 < 20 and we traverse to the left child. The second comparison finds that 16 > 7, so we traverse to the right child. On the third comparison, we succeed.

Each comparison results in reducing the number of items to inspect by one-half. In this respect, the algorithm is similar to a binary search on an array. However, this is true only if the tree is balanced. For example, Figure 3.3 shows another tree containing the same values. While it is a binary search tree, its behavior is more like that of a linked list, with search time increasing proportional to the number of elements stored.

Figure 3.3: An Unbalanced Binary Search Tree

Insertion and Deletion

Let us examine insertions in a binary search tree to determine the conditions that can cause an unbalanced tree. To insert an 18 in the tree in Figure 3.2, we first search for that number. This causes us to arrive at node 16 with nowhere to go. Since 18 > 16, we simply add node 18 to the right child of node 16 (Figure 3.4).

Now we can see how an unbalanced tree can occur. If the data is presented in an ascending sequence, each node will be added to the right of the previous node. This will create one long chain, or linked list. However, if data is presented for insertion in a random order, then a more balanced tree is possible.

Deletions are similar, but require that the binary search tree property be maintained. For example, if node 20 in Figure 3.4 is removed, it must be replaced by node 37. This results in the tree shown in Figure 3.5. The rationale for this choice is as follows. The successor for node 20 must be chosen such that all nodes to the right are larger. Thus, we need to select the smallest valued node to the right of node 20. To make the selection, chain once to the right (node 38), and then chain to the left until the last node is found (node 37). This is the successor for node 20.

Figure 3.4: Binary Tree After Adding Node 18

Figure 3.5: Binary Tree After Deleting Node 20

Implementation

An ANSI-C implementation of a binary search tree may be found in Section 4.6 (pageÊ). Typedef T and comparison operators CompLT and CompEQ should be altered to reflect the data stored in the tree. Each Node consists of Left, Right and Parent pointers designating each child and the parent. Data is stored in the Data field. The tree is based at Root, and is initially NULL. InsertNode allocates a new node and inserts it in the tree. DeleteNode deletes and frees a node from the tree. FindNode searches the tree for a particular value.

a) Red-Black Trees

Binary search trees work best when they are balanced or the path length from root to any leaf is within some bounds. The red-black tree algorithm is a method for balancing trees. The name derives from the fact that each node is colored red or black, and the color of the node is instrumental in determining the balance of the tree. During insert and delete operations, nodes may be rotated to maintain tree balance. Both average and worst-case search time is O(lg n).

This is, perhaps, the most difficult section in the book. If you get glassy-eyed looking at tree rotations, try skipping to skip lists, the next section. For further reading, Cormen[2] has an excellent section on red-black trees.

Theory

A red-black tree is a balanced binary search tree with the following properties[2]:

1. Every node is colored red or black.
2. Every leaf is a NIL node, and is colored black.
3. If a node is red, then both its children are black.
4. Every simple path from a node to a descendant leaf contains the same number of black nodes.

The number of black nodes on a path from root to leaf is known as the black height of a tree. These properties guarantee that any path from the root to a leaf is no more than twice as long as any other. To see why this is true, consider a tree with a black height of two. The shortest distance from root to leaf is two, where both nodes are black. The longest distance from root to leaf is four, where the nodes are colored (root to leaf): red, black, red, black. It is not possible to insert more black nodes as this would violate property 4, the black-height requirement. Since red nodes must have black children (property

3), having two red nodes in a row is not allowed. Thus, the largest path we can construct consists of an alternation of red-black nodes, or twice the length of a path containing only black nodes. All operations on the tree must maintain the properties listed above. In particular, operations which insert or delete items from the tree must abide by these rules.

Insertion

To insert a node, we search the tree for an insertion point, and add the node to the tree. A new node will always be inserted as a leaf node at the bottom of the tree. After insertion, the node is colored red. Then the parent of the node is examined to determine if the red-black tree properties have been violated. If necessary, we recolor the node and do rotations to balance the tree.

By inserting a red node, we have preserved black-height property (property 4). However, property 3 may be violated. This property states that both children of a red node must be black. While both children of the new node are black (they're NIL), consider the case where the parent of the new node is red. Inserting a red node under a red parent would violate this property. There are two cases to consider:

· Red parent, red uncle: Figure 3.6 illustrates a red-red violation. Node X is the newly inserted node, with both parent and uncle colored red. A simple recoloring removes the red-red violation. After recoloring, the grandparent (node B) must be checked for validity, as its parent may be red. Note that this has the effect of propagating a red node up the tree. On completion, the root of the tree is marked black. If it was originally red, then this has the effect of increasing the black-height of the tree.

· Red parent, black uncle: Figure 3.7 illustrates a red-red violation, where the uncle is colored black. Here the nodes may be rotated, with the subtrees adjusted as shown. At this point the algorithm may terminate as there are no red-red conflicts and the top of the subtree (node A) is colored black. Note that if node X was originally a right child, a left rotation would be done first, making the node a left child.

Each adjustment made while inserting a node causes us to travel up the tree one step. At most 1 rotation (2 if the node is a right child) will be done, as the algorithm terminates in this case. The technique for deletion is similar.

Figure 3.6: Insertion Ñ Red Parent, Red Uncle

Figure 3.7: Insertion Ñ Red Parent, Black Uncle

Implementation

An ANSI-C implementation of a red-black tree may be found in Section 4.7 (pageÊ). Typedef T and comparison operators CompLT and CompEQ should be altered to reflect the data stored in the tree. Each Node consists of Left, Right and Parent pointers designating each child and the parent. The node color is stored in Color, and is either Red or Black. The data is stored in the Data field. All leaf nodes of the tree are Sentinel nodes, to simplify coding. The tree is based at Root, and initially is a Sentinel node.

InsertNode allocates a new node and inserts it in the tree. Subsequently, it calls InsertFixup to ensure that the red-black tree properties are maintained. DeleteNode deletes a node from the tree. To maintain red-black tree properties, DeleteFixup is called. FindNode searches the tree for a particular value.

a) Skip Lists

Skip lists are linked lists that allow you to skip to the correct node. Thus the performance bottleneck inherent in a sequential scan is avoided, while insertion and deletion remain relatively efficient. Average search time is O(lg n). Worst-case search time is O(n), but is extremely unlikely. An excellent reference for skip lists is Pugh[5].

Theory

The indexing scheme employed in skip lists is similar in nature to the method used to lookup names in an address book. To lookup a name, you index to the tab representing the first character of the desired entry. In Figure 3.8, for example, the top-most list represents a simple linked list with no tabs. Adding tabs (middle figure) facilitates the search. In this case, level-1 pointers are traversed. Once the correct segment of the list is found, level-0 pointers are traversed to find the specific entry.

The indexing scheme may be extended as shown in the bottom figure, where we now have an index to the index. To locate an item, level-2 pointers are traversed until the correct segment of the list is identified. Subsequently, level-1 and level-0 pointers are traversed.

During insertion the number of pointers required for a new node must be determined. This is easily resolved using a probabilistic technique. A random number generator is used to toss a computer coin. When inserting a new node, the coin is tossed to determine if it should be level-1. If you win, the coin is tossed again to determine if the node should be level-2. Another win, and the coin is tossed to determine if the node should be level-3. This process repeats until you lose.

The skip list algorithm has a probabilistic component, and thus has a probabilistic bounds on the time required to execute. However, these bounds are quite tight in normal circumstances. For example, to search a list containing 1000 items, the probability that search time will be 5 times the average is about 1 in 1,000,000,000,000,000,000[5].

Figure 3.8: Skip List Construction

Implementation

An ANSI-C implementation of a skip list may be found in Section 4.8 (pageÊ). Typedef T and comparison operators CompLT and CompEQ should be altered to reflect the data stored in the list. In addition, MAXLEVEL should be set based on the maximum size of the dataset.

To initialize, InitList is called. The list header is allocated and initialized. To indicate an empty list, all levels are set to point to the header. InsertNode allocates a new node and inserts it in the list. InsertNode first searches for the correct insertion point. While searching, the update array maintains pointers to the upper-level nodes encountered. This information is subsequently used to establish correct links for the newly inserted node. NewLevel is determined using a random number generator, and the node allocated. The forward links are then established using information from the update array.

DeleteNode deletes and frees a node, and is implemented in a similar manner. FindNode searches the list for a particular value.

a) Comparison of Methods

We have seen several ways to construct dictionaries: hash tables, unbalanced binary search trees, red-black trees and skip lists. There are several factors that influence the choice of an algorithm:

· Sorted output. If sorted output is required, then hash tables are not a viable alternative. Entries are stored in the table based on their hashed value, with no other ordering. For binary trees, the story is different. An in-order tree walk will produce a sorted list. For example:

```
void WalkTree(Node *P) {
    if (P == NIL) return;
    WalkTree(P->Left);

    /* examine P->Data here */

    WalkTree(P->Right);
}
WalkTree(Root);
```

· To examine skip list nodes in order, simply chain through the level-0 pointers. For example:

```
Node *P = List.Hdr->Forward[0];
while (P != NIL) {

    /* examine P->Data here */

    P = P->Forward[0];
}
```

· Space. The amount of memory required to store a value should be minimized. This is especially true if many small nodes are to be allocated.

¨ For hash tables, only one forward pointer per node is required. In addition, the hash table itself must be allocated.

¨ For red-black trees, each node has a left, right and parent pointer. In addition, the color of each node must be recorded. Although this requires only one bit, more space may be allocated to ensure that the size of the structure is properly aligned. Thus, each node in a red-black tree requires enough space for 3-4 pointers.

¨ For skip lists, each node has a level-0 forward pointer. The probability of having a level-1 pointer is 1 2. The probability of having a level-2 pointer is 1 4. In general, the number of forward pointers per node is

¨

· Time. The algorithm should be efficient. This is especially true if a large dataset is expected. Table 3.2 compares the search time for each algorithm. Note that worst-case behavior for hash tables and skip lists is extremely unlikely. Actual timing tests are described below.

· Simplicity. If the algorithm is short and easy to understand, fewer mistakes may be made. This not only makes your life easy, but the maintenance programmer entrusted with the task of making repairs will appreciate any efforts you make in this area. The number of statements required for each algorithm is listed in Table 3.2.

method statements average time worst-case time hash table 26 O(1) O(n) unbalanced tree 41 O(lg n) O(n) red-black tree 120 O(lg n) O(lg n) skip list 55 O(lg n) O(n) Table 3.2: Comparison of Dictionaries

Average time for insert, search and delete operations on a database of 65,536 (216) randomly input items may be found in Table 3.3. For this test the hash table size was 10,009 and 16 index levels were allowed for the skip list. While there is some variation in the timings for the four methods, they are close enough so that other considerations should come into play when selecting an algorithm.

method insert search delete hash table 18 8 10 unbalanced tree 37 17 26 red-black tree 40 16 37 skip list 48 31 35 Table 3.3: Average Time (ms), 65536 Items, Random Input

Table 3.4 shows the average search time for two sets of data: a random set, where all values are unique, and an ordered set, where values are in ascending order. Ordered input creates a worst-case scenario for unbalanced tree algorithms, as the tree ends up being a simple linked list. The times shown are for a single search operation. If we were to search for all items in a database of 65,536 values, a red-black tree algorithm would take .6Êseconds, while an unbalanced tree algorithm would take 1 hour.

count hash table unbalanced tree red-black tree skip list 16 4 3 2 5 random 256 3 4 4 9 input 4,096 3 7 6 12 65,536 8 17 16 31 16 3 4 2 4 ordered 256 3 47 4 7 input 4,096 3 1,033 6 11 65,536 7 55,019 9 15 Table 3.4: Average Search Time (us)

1. Code Listings

a) Insertion Sort Code

```
typedef int T;
typedef int TblIndex;

#define CompGT(a,b) (a > b)

void InsertSort(T *Lb, T *Ub) {
    T V, *I, *J, *Jmin;

   /************************
    *  Sort Array[Lb..Ub]  *
    ************************/
    Jmin = Lb - 1;
    for (I = Lb + 1; I <= Ub; I++) {
        V = *I;

        /* Shift elements down until       */
        /* insertion point found.          */
        for (J = I-1; J != Jmin && CompGT(*J, V); J--)
            *(J+1) = *J;
        *(J+1) = V;
    }
}
```

a) Shell Sort Code

```
typedef int T;
typedef int TblIndex;

#define CompGT(a,b) (a > b)

void ShellSort(T *Lb, T *Ub) {
    TblIndex H, N;
    T V, *I, *J, *Min;

   /*************************
    *  Sort array A[Lb..Ub]  *
    *************************/

    /* compute largest increment */
    N = Ub - Lb + 1;
    H = 1;
    if (N < 14)
        H = 1;
    else if (sizeof(TblIndex) == 2 && N > 29524)
        H = 3280;
    else {
        while (H < N) H = 3*H + 1;
        H /= 3;
        H /= 3;
    }

    while (H > 0) {

        /* sort-by-insertion in increments of H */
        /* Care must be taken for pointers that */
        /* wrap through zero.                   */
        Min = Lb + H;
        for (I = Min; I <= Ub; I++) {
            V = *I;
            for (J = I-H; CompGT(*J, V); J -= H) {
                *(J+H) = *J;
                if (J <= Min) { J -= H; break; }
            }
            *(J+H) = V;
        }

        /* compute next increment */
        H /= 3;
    }
}
```

a) Quicksort Code

```
typedef int T;
typedef int TblIndex;

#define CompGT(a,b) (a > b)

T *Partition(T *Lb, T *Ub) {
    T V, Pivot, *I, *J, *P;
    unsigned int Offset;

   /*****************************
    *  partition Array[Lb..Ub]  *
    *****************************/

    /* select pivot and exchange with 1st element */
    Offset = (Ub - Lb)>>1;
    P = Lb + Offset;
    Pivot = *P;
    *P = *Lb;

    I = Lb + 1;
    J = Ub;
    while (1) {
        while (I < J && CompGT(Pivot, *I)) I++;
        while (J >= I && CompGT(*J, Pivot)) J--;
        if (I >= J) break;
        V = *I;
        *I = *J;
        *J = V;
        J--; I++;
    }

    /* pivot belongs in A[j] */
    *Lb = *J;
    *J = Pivot;

    return J;
}

void QuickSort(T *Lb, T *Ub) {
    T *M;

   /*************************
    *  Sort array A[Lb..Ub]  *
    *************************/

    while (Lb < Ub) {

        /* quickly sort short lists */
        if (Ub - Lb <= 12) {
            InsertSort(Lb, Ub);
            return;
        }

        /* partition into two segments */
        M = Partition (Lb, Ub);

        /* sort the smallest partition    */
        /* to minimize stack requirements */
```

```
        if (M - Lb <= Ub - M) {
            QuickSort(Lb, M - 1);
            Lb = M + 1;
        } else {
            QuickSort(M + 1, Ub);
            Ub = M - 1;
        }
    }
}
```

a) Qsort Code

```
#include <limits.h>
#define MAXSTACK (sizeof(size_t) * CHAR_BIT)

static void Exchange(void *a, void *b, size_t size) {
    size_t i;

    /******************
     *  exchange a,b  *
     ******************/

    for (i = sizeof(int); i <= size; i += sizeof(int)) {
        int t = *((int *)a);
        *(((int *)a)++) = *((int *)b);
        *(((int *)b)++) = t;
    }
    for (i = i - sizeof(int) + 1; i <= size; i++) {
        char t = *((char *)a);
        *(((char *)a)++) = *((char *)b);
        *(((char *)b)++) = t;
    }
}

void qsort(void *base, size_t nmemb, size_t size,
        int (*compar)(const void *, const void *)) {
    void *LbStack[MAXSTACK], *UbStack[MAXSTACK];
    int sp;
    unsigned int Offset;

    /********************
     *  ANSI-C qsort()  *
     ********************/

    LbStack[0] = (char *)base;
    UbStack[0] = (char *)base + (nmemb-1)*size;
    for (sp = 0; sp >= 0; sp--) {
        char *Lb, *Ub, *M;
        char *P, *I, *J;

        Lb = LbStack[sp];
        Ub = UbStack[sp];

        while (Lb < Ub) {

            /* select pivot and exchange with 1st element */
            Offset = (Ub - Lb) >> 1;
            P = Lb + Offset - Offset % size;
            Exchange (Lb, P, size);

            /* partition into two segments */
            I = Lb + size;
            J = Ub;
            while (1) {
                while (I < J && compar(Lb, I) > 0) I += size;
                while (J >= I && compar(J, Lb) > 0) J -= size;
                if (I >= J) break;
                Exchange (I, J, size);
                J -= size;
                I += size;
```

```
            }

            /* pivot belongs in A[j] */
            Exchange (Lb, J, size);
            M = J;

            /* keep processing smallest segment, and stack largest */
            if (M - Lb <= Ub - M) {
                if (M + size < Ub) {
                    LbStack[sp] = M + size;
                    UbStack[sp++] = Ub;
                }
                Ub = M - size;
            } else {
                if (M - size > Lb) {
                    LbStack[sp] = Lb;
                    UbStack[sp++] = M - size;
                }
                Lb = M + size;
            }
        }
    }
}
```

a) Hash Table Code

```
#include <stdlib.h>
#include <stdio.h>

/* modify these lines to establish data type */
typedef int T;
#define CompEQ(a,b) (a == b)

typedef struct Node_ {
    struct Node_ *Next;         /* next node */
    T Data;                     /* data stored in node */
} Node;

typedef int HashTableIndex;

Node **HashTable;
int HashTableSize;

HashTableIndex Hash(T Data) {
    /* division method */
    return (Data % HashTableSize);
}

Node *InsertNode(T Data) {
    Node *p, *p0;
    HashTableIndex bucket;

    /* insert node at beginning of list */
    bucket = Hash(Data);
    if ((p = malloc(sizeof(Node))) == 0) {
        fprintf (stderr, "out of memory (InsertNode)\n");
        exit(1);
    }
    p0 = HashTable[bucket];
    HashTable[bucket] = p;
    p->Next = p0;
```

```
    p->Data = Data;
    return p;
}

void DeleteNode(T Data) {
    Node *p0, *p;
    HashTableIndex bucket;

    /* find node */
    p0 = 0;
    bucket = Hash(Data);
    p = HashTable[bucket];
    while (p && !CompEQ(p->Data, Data)) {
        p0 = p;
        p = p->Next;
    }
    if (!p) return;

    /* p designates node to delete, remove it from list */
    if (p0)
        /* not first node, p0 points to previous node */
        p0->Next = p->Next;
    else
        /* first node on chain */
        HashTable[bucket] = p->Next;

    free (p);
}

Node *FindNode (T Data) {
    Node *p;
    p = HashTable[Hash(Data)];
    while (p && !CompEQ(p->Data, Data))
        p = p->Next;
    return p;
}
```

a) Binary Search Tree Code

```
#include <stdio.h>
#include <stdlib.h>

/* modify these lines to establish data type */
typedef int T;
#define CompLT(a,b) (a < b)
#define CompEQ(a,b) (a == b)

typedef struct Node_ {
    struct Node_ *Left;         /* left child */
    struct Node_ *Right;        /* right child */
    struct Node_ *Parent;       /* parent */
    T Data;                     /* data stored in node */
} Node;

Node *Root = NULL;               /* root of binary tree */

Node *InsertNode(T Data) {
    Node *X, *Current, *Parent;

   /*********************************************
    *  allocate node for Data and insert in tree  *
    *********************************************/
```

```
    /* setup new node */
    if ((X = malloc (sizeof(*X))) == 0) {
        fprintf (stderr, "insufficient memory (InsertNode)\n");
        exit(1);
    }
    X->Data = Data;
    X->Left = NULL;
    X->Right = NULL;

    /* find X's parent */
    Current = Root;
    Parent = 0;
    while (Current) {
        if (CompEQ(X->Data, Current->Data)) return (Current);
        Parent = Current;
        Current = CompLT(X->Data, Current->Data) ?
            Current->Left : Current->Right;
    }
    X->Parent = Parent;

    /* insert X in tree */
    if(Parent)
        if(CompLT(X->Data, Parent->Data))
            Parent->Left = X;
        else
            Parent->Right = X;
    else
        Root = X;

    return(X);
}
void DeleteNode(Node *Z) {
    Node *X, *Y;

   /*****************************
    *  delete node Z from tree  *
    *****************************/

    /* Y will be removed from the parent chain */
    if (!Z || Z == NULL) return;

    /* find tree successor */
    if (Z->Left == NULL || Z->Right == NULL)
        Y = Z;
    else {
        Y = Z->Right;
        while (Y->Left != NULL) Y = Y->Left;
    }

    /* X is Y's only child */
    if (Y->Left != NULL)
        X = Y->Left;
    else
        X = Y->Right;

    /* remove Y from the parent chain */
    if (X) X->Parent = Y->Parent;
    if (Y->Parent)
        if (Y == Y->Parent->Left)
            Y->Parent->Left = X;
```

```
        else
            Y->Parent->Right = X;
    else
        Root = X;

    /* Y is the node we're removing */
    /* Z is the data we're removing */
    /* if Z and Y are not the same, replace Z with Y. */
    if (Y != Z) {
        Y->Left = Z->Left;
        if (Y->Left) Y->Left->Parent = Y;
        Y->Right = Z->Right;
        if (Y->Right) Y->Right->Parent = Y;
        Y->Parent = Z->Parent;
        if (Z->Parent)
            if (Z == Z->Parent->Left)
                Z->Parent->Left = Y;
            else
                Z->Parent->Right = Y;
        else
            Root = Y;
        free (Z);
    } else {
        free (Y);
    }
}

Node *FindNode(T Data) {

   /******************************
    *  find node containing Data  *
    ******************************/

    Node *Current = Root;
    while(Current != NULL)
        if(CompEQ(Data, Current->Data))
            return (Current);
        else
            Current = CompLT (Data, Current->Data) ?
                Current->Left : Current->Right;
    return(0);
}
```

a) Red-Black Tree Code

```
#include <stdlib.h>
#include <stdio.h>

/* modify these lines to establish data type */
typedef int T;
#define CompLT(a,b) (a < b)
#define CompEQ(a,b) (a == b)

/* red-black tree description */
typedef enum { Black, Red } NodeColor;

typedef struct Node_ {
    struct Node_ *Left;         /* left child */
    struct Node_ *Right;        /* right child */
    struct Node_ *Parent;       /* parent */
    NodeColor Color;            /* node color (black, red) */
```

```
    T Data;                     /* data stored in node */
} Node;

#define NIL &Sentinel           /* all leafs are sentinels */
Node Sentinel = { NIL, NIL, 0, Black, 0};

Node *Root = NIL;               /* root of red-black tree */


Node *InsertNode(T Data) {
    Node *Current, *Parent, *X;

   /***********************************************
    *  allocate node for Data and insert in tree  *
    ***********************************************/

    /* setup new node */
    if ((X = malloc (sizeof(*X))) == 0) {
        printf ("insufficient memory (InsertNode)\n");
        exit(1);
    }
    X->Data = Data;
    X->Left = NIL;
    X->Right = NIL;
    X->Parent = 0;
    X->Color = Red;

    /* find where node belongs */
    Current = Root;
    Parent = 0;
    while (Current != NIL) {
        if (CompEQ(X->Data, Current->Data)) return (Current);
        Parent = Current;
        Current = CompLT(X->Data, Current->Data) ?
            Current->Left : Current->Right;
    }

    /* insert node in tree */
    if(Parent) {
        if(CompLT(X->Data, Parent->Data))
            Parent->Left = X;
        else
            Parent->Right = X;
        X->Parent = Parent;
    } else
        Root = X;

    InsertFixup(X);
    return(X);
}

void InsertFixup(Node *X) {

   /*************************************
    *  maintain red-black tree balance  *
    *  after inserting node X           *
    *************************************/

    /* check red-black properties */
    while (X != Root && X->Parent->Color == Red) {
        /* we have a violation */
```

```
        if (X->Parent == X->Parent->Parent->Left) {
            Node *Y = X->Parent->Parent->Right;
            if (Y->Color == Red) {

                /* uncle is red */
                X->Parent->Color = Black;
                Y->Color = Black;
                X->Parent->Parent->Color = Red;
                X = X->Parent->Parent;
            } else {

                /* uncle is black */
                if (X == X->Parent->Right) {
                    /* make X a left child */
                    X = X->Parent;
                    RotateLeft(X);
                }

                /* recolor and rotate */
                X->Parent->Color = Black;
                X->Parent->Parent->Color = Red;
                RotateRight(X->Parent->Parent);
            }
        } else {

            /* mirror image of above code */
            Node *Y = X->Parent->Parent->Left;
            if (Y->Color == Red) {

                /* uncle is red */
                X->Parent->Color = Black;
                Y->Color = Black;
                X->Parent->Parent->Color = Red;
                X = X->Parent->Parent;
            } else {

                /* uncle is black */
                if (X == X->Parent->Left) {
                    X = X->Parent;
                    RotateRight(X);
                }
                X->Parent->Color = Black;
                X->Parent->Parent->Color = Red;
                RotateLeft(X->Parent->Parent);
            }
        }
    }
    Root->Color = Black;
}

void RotateLeft(Node *X) {

   /************************
    *  rotate Node X to left *
    ************************/

    Node *Y = X->Right;

    /* establish X->Right link */
    X->Right = Y->Left;
    if (Y->Left != NIL) Y->Left->Parent = X;
```

```
    /* establish Y->Parent link */
    if (Y != NIL) Y->Parent = X->Parent;
    if (X->Parent) {
        if (X == X->Parent->Left)
            X->Parent->Left = Y;
        else
            X->Parent->Right = Y;
    } else {
        Root = Y;
    }

    /* link X and Y */
    Y->Left = X;
    if (X != NIL) X->Parent = Y;
}

void RotateRight(Node *X) {

   /****************************
    *  rotate Node X to right  *
    ****************************/

    Node *Y = X->Left;

    /* establish X->Left link */
    X->Left = Y->Right;
    if (Y->Right != NIL) Y->Right->Parent = X;

    /* establish Y->Parent link */
    if (Y != NIL) Y->Parent = X->Parent;
    if (X->Parent) {
        if (X == X->Parent->Right)
            X->Parent->Right = Y;
        else
            X->Parent->Left = Y;
    } else {
        Root = Y;
    }

    /* link X and Y */
    Y->Right = X;
    if (X != NIL) X->Parent = Y;
}

void DeleteNode(Node *Z) {
    Node *X, *Y;

   /*****************************
    *  delete node Z from tree  *
    *****************************/

    if (!Z || Z == NIL) return;

    if (Z->Left == NIL || Z->Right == NIL) {
        /* Y has a NIL node as a child */
        Y = Z;
    } else {
        /* find tree successor with a NIL node as a child */
        Y = Z->Right;
        while (Y->Left != NIL) Y = Y->Left;
```

```
    }

    /* X is Y's only child */
    if (Y->Left != NIL)
        X = Y->Left;
    else
        X = Y->Right;

    /* remove Y from the parent chain */
    X->Parent = Y->Parent;
    if (Y->Parent)
        if (Y == Y->Parent->Left)
            Y->Parent->Left = X;
        else
            Y->Parent->Right = X;
    else
        Root = X;

    if (Y != Z) Z->Data = Y->Data;
    if (Y->Color == Black)
        DeleteFixup (X);
    free (Y);
}

void DeleteFixup(Node *X) {

   /************************************
    *  maintain red-black tree balance  *
    *  after deleting node X            *
    ************************************/

    while (X != Root && X->Color == Black) {
        if (X == X->Parent->Left) {
            Node *W = X->Parent->Right;
            if (W->Color == Red) {
                W->Color = Black;
                X->Parent->Color = Red;
                RotateLeft (X->Parent);
                W = X->Parent->Right;
            }
            if (W->Left->Color == Black && W->Right->Color == Black) {
                W->Color = Red;
                X = X->Parent;
            } else {
                if (W->Right->Color == Black) {
                    W->Left->Color = Black;
                    W->Color = Red;
                    RotateRight (W);
                    W = X->Parent->Right;
                }
                W->Color = X->Parent->Color;
                X->Parent->Color = Black;
                W->Right->Color = Black;
                RotateLeft (X->Parent);
                X = Root;
            }
        } else {
            Node *W = X->Parent->Left;
            if (W->Color == Red) {
                W->Color = Black;
                X->Parent->Color = Red;
```

```
                RotateRight (X->Parent);
                W = X->Parent->Left;
            }
            if (W->Right->Color == Black && W->Left->Color == Black) {
                W->Color = Red;
                X = X->Parent;
            } else {
                if (W->Left->Color == Black) {
                    W->Right->Color = Black;
                    W->Color = Red;
                    RotateLeft (W);
                    W = X->Parent->Left;
                }
                W->Color = X->Parent->Color;
                X->Parent->Color = Black;
                W->Left->Color = Black;
                RotateRight (X->Parent);
                X = Root;
            }
        }
    }
    X->Color = Black;
}

Node *FindNode(T Data) {

   /*******************************
    *  find node containing Data  *
    *******************************/

    Node *Current = Root;
    while(Current != NIL)
        if(CompEQ(Data, Current->Data))
            return (Current);
        else
            Current = CompLT (Data, Current->Data) ?
                Current->Left : Current->Right;
    return(0);
}
```

a) Skip List Code

```
#include <stdio.h>
#include <stdlib.h>

/* define data-type and compare operators here */
typedef int T;
#define CompLT(a,b) (a < b)
#define CompEQ(a,b) (a == b)

/*
 * levels range from (0 .. MAXLEVEL)
 */
#define MAXLEVEL 15

typedef struct Node_ {
    T Data;                     /* user's data */
    struct Node_ *Forward[1];   /* skip list forward pointer */
} Node;

typedef struct {
```

```
    Node *Hdr;                  /* List Header */
    int ListLevel;              /* current level of list */
} SkipList;

SkipList List;                  /* skip list information */

#define NIL List.Hdr

void InitList() {
    int i;

   /*************************
    *  initialize skip list  *
    *************************/

    if ((List.Hdr = malloc(sizeof(Node) + MAXLEVEL*sizeof(Node *))) == 0) {
        printf ("insufficient memory (InitList)\n");
        exit(1);
    }
    for (i = 0; i <= MAXLEVEL; i++)
        List.Hdr->Forward[i] = NIL;
    List.ListLevel = 0;
}

Node *InsertNode(T Data) {
    int i, NewLevel;
    Node *update[MAXLEVEL+1];
    Node *X;

   /*********************************************
    *  allocate node for Data and insert in list  *
    *********************************************/

    /* find where data belongs */
    X = List.Hdr;
    for (i = List.ListLevel; i >= 0; i--) {
        while (X->Forward[i] != NIL
          && CompLT(X->Forward[i]->Data, Data))
            X = X->Forward[i];
        update[i] = X;
    }
    X = X->Forward[0];
    if (X != NIL && CompEQ(X->Data, Data)) return(X);

    /* determine level */
    NewLevel = 0;
    while (rand() < RAND_MAX/2) NewLevel++;
    if (NewLevel > MAXLEVEL) NewLevel = MAXLEVEL;

    if (NewLevel > List.ListLevel) {
        for (i = List.ListLevel + 1; i <= NewLevel; i++)
            update[i] = NIL;
        List.ListLevel = NewLevel;
    }

    /* make new node */
    if ((X = malloc(sizeof(Node) +
      NewLevel*sizeof(Node *))) == 0) {
        printf ("insufficient memory (InsertNode)\n");
        exit(1);
    }
```

```
    X->Data = Data;

    /* update forward links */
    for (i = 0; i <= NewLevel; i++) {
        X->Forward[i] = update[i]->Forward[i];
        update[i]->Forward[i] = X;
    }
    return(X);
}

void DeleteNode(T Data) {
    int i;
    Node *update[MAXLEVEL+1], *X;

   /****************************************
    *  delete node containing Data from list  *
    ****************************************/

    /* find where data belongs */
    X = List.Hdr;
    for (i = List.ListLevel; i >= 0; i--) {
        while (X->Forward[i] != NIL
          && CompLT(X->Forward[i]->Data, Data))
            X = X->Forward[i];
        update[i] = X;
    }
    X = X->Forward[0];
    if (X == NIL || !CompEQ(X->Data, Data)) return;

    /* adjust forward pointers */
    for (i = 0; i <= List.ListLevel; i++) {
        if (update[i]->Forward[i] != X) break;
        update[i]->Forward[i] = X->Forward[i];
    }

    free (X);

    /* adjust header level */
    while ((List.ListLevel > 0)
    && (List.Hdr->Forward[List.ListLevel] == NIL))
        List.ListLevel--;
}

Node *FindNode(T Data) {
    int i;
    Node *X = List.Hdr;

   /******************************
    *  find node containing Data  *
    ******************************/

    for (i = List.ListLevel; i >= 0; i--) {
        while (X->Forward[i] != NIL
          && CompLT(X->Forward[i]->Data, Data))
            X = X->Forward[i];
    }
    X = X->Forward[0];
    if (X != NIL && CompEQ(X->Data, Data)) return (X);
    return(0);
}
```

1. Bibliography

[1] Donald E. Knuth. The Art of Computer Programming, volume 3. Massachusetts:
Addison-Wesley, 1973.

[2] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. Introduction to Algo-
rithms. New York: McGraw-Hill, 1992.

[3] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. Data Structures and Algorithms. Massachusetts: Addison-Wesley, 1983.

[4] Peter K. Pearson. Fast hashing of variable-length text strings. Communications of the
ACM, 33(6):677-680, June 1990.

[5] William Pugh. Skip lists: A probabilistic alternative to balanced trees. Communications
of the ACM, 33(6):668-676, June 1990.

v €

v v ¦ ‰ r ~ ‰ r ˆ ‰ r ÿ

ÿÿÿÿ r ‰ - ÿÿÿÿÿÿ
‰ r
t - < < < p - < < <
¡ d

PPNT " Arial ,

Helvetica

. +' 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t -) <)) < <
) p -) <)) < <) ¡ d

PPNT " Arial * 7 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t -) ; < ; ; <) <) ; p -) ; < ; ; <) <) ; ¡ d

PPNT " Arial (5 $ 16 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ; M
< M M < ; < ; M p - ; M < M M < ; < ; M ¡ d

PPNT " Arial * 20 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - M _ < _
_ < M < M _ p - M _ < _ _ < M < M _ ¡ d

PPNT " Arial * 37 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - _ q < q
q < _ < _ q p - _ q < q q < _ < _ q ¡ d

PPNT " Arial * 38 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - q *f* < *f* *f* < q < q *f* p - q *f* < *f* *f* < q < q *f* ¡ d

PPNT " Arial * 43 ¡ d PPNT " System ¡ d

PPNT Arial (0 ¡ d PPNT " System ¡ d

PPNT Arial * 1 ¡ d PPNT " System ¡ d

PPNT Arial * 2 ¡ d PPNT " System ¡ d

PPNT Arial * 3 ¡ d PPNT " System ¡ d

PPNT Arial * 4 ¡ d PPNT " System ¡ d

PPNT Arial * 5 ¡ d PPNT " System ¡ d

PPNT Arial * 6 ¡ d PPNT " System p
@
S
@
G
S ÿÿÿÿÿÿ
ÿÿÿÿÿÿÿÿ t N
< B
<
B
A
A

A

A
A
A
A

A

```
A   A   A   A   A   B
< p   D @ D S D @ D G D S t N A < G B D < A B A A B A B A C A C A C A D A D A E
A E A F A F A F A G B D < p   z @ z S z @ z G z S t N w < } B z < w B x A x A y
A y A y A z A z A { A { A { A | A | A } A } B z < ¡ d
```

PPNT Arial)N Ub ¡ d PPNT " System ¡ d

PPNT Arial (G X M ¡ d PPNT " System ¡ d

PPNT
Arial (V Lb ¡ d PPNT " System ÿ46 " e e
e ÿ

ÿÿÿÿ e - ÿÿÿÿÿÿ

e

t - p ~ ~ ~ p p ~ p - p ~ ~ ~ p p ~

ÿÿÿÿÿÿ t - ~ ‹ ‹ ‹ ~ ~ ‹ p - ~ ‹ ‹ ‹ ~ ~ ‹

¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial ,

Helvetica

. +

‡ 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - p 8 ~ S ~ 8 ~ S p S p 8 ~ 8
p - p 8 ~ S ~ 8 ~ S p S p 8 ~ 8 ÿÿÿÿÿÿ t - ~ 8 ‹ S ‹ 8 ‹ S ~ S ~ 8 ‹ 8
p - ~ 8 ‹ S ‹ 8 ‹ S ~ S ~ 8 ‹ 8 ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)6 7 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - p n ~ Š ~ n ~ Š p Š p n ~ n p - p n ~ Š ~ n ~ Š p Š p n ~ n ÿÿÿÿÿÿ t - ~ n ‹ Š ‹ n ‹ Š ~ Š ~ n ‹ n p - ~ n ‹ Š ‹ n ‹ Š ~ Š ~ n ‹ n ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)4 16 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - p ¥ ~ À ~
¥ ~ À p À p ¥ ~ ¥ p - p ¥ ~ À ~ ¥ ~ À p À p ¥ ~ ¥ ÿÿÿÿÿÿ t - ~ ¥ ‹ À ‹
¥ ‹ À ~ À ~ ¥ ‹ ¥ p - ~ ¥ ‹ À ‹ ¥ ‹ À ~ À ~ ¥ ‹ ¥ ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)7 20 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - p Û ~ ÷ ~
Û ~ ÷ p ÷ p Û ~ Û p - p Û ~ ÷ ~ Û ~ ÷ p ÷ p Û ~ Û ÿÿÿÿÿÿ t - ~ Û ‹ ÷ ‹
Û ‹ ÷ ~ ÷ ~ Û ‹ Û p - ~ Û ‹ ÷ ‹ Û ‹ ÷ ~ ÷ ~ Û ‹ Û ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)6 37 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - p ~ -
~ ~ - p - p ~ p - p ~ - ~ ~ - p - p ~ ÿÿÿÿÿÿ t - ~ ‹ -
‹ ‹ - ~ - ~ ‹ p - ~ ‹ - ‹ ‹ - ~ - ~ ‹ ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)6 38 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - p H ~ c ~ H ~ c p c p H ~ H p - p H ~ c ~ H ~ c p c p H ~ H ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial

(z S # ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ~ H ‹ c ‹ H ‹ c ~ c ~ H ‹ H
p - ~ H ‹ c ‹ H ‹ c ~ c ~ H ‹ H ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial
(‡ Q 43 ¡ d PPNT " System p w w 4 w w 4 ÿÿÿÿÿÿ
ÿÿÿÿÿÿÿÿ t ¾ u
y w w v v v v u u u u u u u u u

u

u

v
v
w
w
w
x
x
x

x

x

y

y

y y y y y y y x x x x x w w t N t 2 z 8 w 8 z 2 y 3
y 3 y 3 x 3 x 3 w 3 w 3 w 3 v 3 v 3 u 3 u 3 u 3 t 2 w 8 ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p w F w j w F w j t ¾ u D
y H w H w H v H v H v G v G u G u G u G u F u F u F u E u E u E u E u D v D v D
v D v D w D w D w D x D x D x D x D x D y E y E y E y E y F y F y F y G y G x G
x G x G x H x H w H w H t N t i z n w n z i y i y i y i x i x i w j w j w j v i
v i u i u i u i t i w n ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p w | w ¡ w | w ¡ t ¾ u z
y ~ w ~ w ~ v ~ v ~ v ~ v ~ u } u } u } u } u | u | u | u | u { u { u { v { v z
v z v z w z w z w z x z x z x z x { x { y { y { y | y | y | y | y } y } y } x }
x ~ x ~ x ~ x ~ w ~ w ~ t N t Ÿ z ¥ w ¥ z Ÿ y Ÿ y y x x w w w v
v u u u Ÿ t Ÿ w ¥ ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p w ³ w × w ³ w × t ¾ u °
y µ w µ w µ v ´ v ´ v ´ v ´ u ´ u ´ u ³ u ³ u ³ u ³ u ² u ² u ² u ± u ± v ± v ±
v ± v ± w ° w ° w ° x ± x ± x ± x ± x ± y ± y ² y ² y ² y ³ y ³ y ³ y ³ y ´ x ´
x ´ x ´ x ´ x ´ w µ w µ t N t Ö z Û w Û z Ö y Ö y Ö y Ö x Ö x Ö w Ö w Ö w Ö v Ö
v Ö u Ö u Ö u Ö t Ö w Û ¡ d

PPNT Arial ¡ d PPNT " System ¡ d PPNT " System p w é w
w é w
t ¾ u ç y ë w ë w ë v ë v ë v ë v ê u ê u ê u ê u é u é u é u é u è u è u è u è
v ç v ç v ç v ç w ç w ç w ç x ç x ç x ç x ç x è y è y è y è y é y é y é y é y ê
y ê x ê x ê x ë x ë x ë w ë w ë t N t

z w z

y

y

y
x
x
w
w
w
v
v
u
u

u

t

w ¡ d

PPNT Arial ¡ d PPNT " System ¡ d PPNT " System p w w D w -
w D t ¾ u y ! w ! w ! v ! v ! v ! v ! u ! u u u u u u u -
u - u - u - v - v - v v w w w x x x - x - x - y - y - y y y y
y y y
x ! x ! x ! x ! x ! w ! w ! t N t C z H w H z C y C y C y C x C x C w C w C w C
v C v C u C u C u C t C w H ¡ d

PPNT Arial ¡ d PPNT " System ¡ d PPNT " System ÿÿÿÿÿÿ
t - # Š 1 ¥ 1 Š 1 ¥ # ¥ # Š 1 Š p - # Š 1 ¥ 1 Š 1 ¥ # ¥ # Š 1 Š
¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial (-
’ 18 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - Š # ¥ # Š # ¥ ¥ Š # Š
p - Š # ¥ # Š # ¥ ¥ Š # Š ¡ d PPNT 1
Courier New ¡ d PPNT " System ¡ d PPNT 1
Courier New ,
Courier

(Y X ¡ d PPNT " System ¡ d PPNT 1
Courier New ¡ d PPNT " System ¡ d PPNT 1
Courier New + + P ¡ d PPNT " System p P | k | P | ` | k | ÿÿÿÿÿÿ
ÿÿÿÿÿÿÿÿ t k y p k p | k y k p j … j z … t „ -
Š „ Š „ „ ÿ7m

7 ¥ 7 ¥ð 7 ¥ ÿ

ÿÿÿÿ ¥ 7 - ÿÿÿÿÿÿ
7 ¥
t - ¦ % Ê % ¦ % Ê Ê ¦ % ¦ p - ¦ % Ê % ¦ % Ê Ê ¦ % ¦
¡ d

PPNT " Arial ,

Helvetica

. +µ-

4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - % ¦ 8 Ê 8 ¦ 8 Ê % Ê % ¦ 8 ¦
p - % ¦ 8 Ê 8 ¦ 8 Ê % Ê % ¦ 8 ¦ ÿÿÿÿÿÿ t - 8 ¦ J Ê J ¦ J Ê 8 Ê 8 ¦ J ¦
p - 8 ¦ J Ê J ¦ J Ê 8 Ê 8 ¦ J ¦ ¡ d

PPNT " Arial *$ 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - J ¦ \ Ê \
¦ \ Ê J Ê J ¦ \ ¦ p - J ¦ \ Ê \ ¦ \ Ê J Ê J ¦ \ ¦ ¡ d

PPNT " Arial * 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - 9 %] % 9 %]] 9 % 9 p - 9 %] % 9 %]] 9 % 9 ¡ d

PPNT " Arial (-
H 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - % 9 8] 8 9 8] %] % 9 8 9
p - % 9 8] 8 9 8] %] % 9 8 9 ¡ d

PPNT " Arial * 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - 8 9 J] J
9 J] 8] 8 9 J 9 p - 8 9 J] J 9 J] 8] 8 9 J 9 ¡ d

PPNT " Arial * 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - J 9 \ ] \ 9 \ ] J] J 9 \ 9 p - J 9 \ ] \ 9 \ ] J] J 9 \ 9 ¡ d

PPNT " Arial * 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - % 7 %
% 7 7 % p - % 7 % % 7 7 % ÿÿÿÿÿÿ t - % 8 7 8
8 7 % 7 % 8 p - % 8 7 8 8 7 % 7 % 8 ¡ d

PPNT " Arial (1 " 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - 8 J 7
J J 7 8 7 8 J p - 8 J 7 J J 7 8 7 8 J ¡ d

PPNT " Arial * 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - J \ 7 \
\ 7 J 7 J \ p - J \ 7 \ \ 7 J 7 J \ ¡ d

PPNT " Arial * 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - % £ %
% £ £ % p - % £ % % £ £ % ¡ d

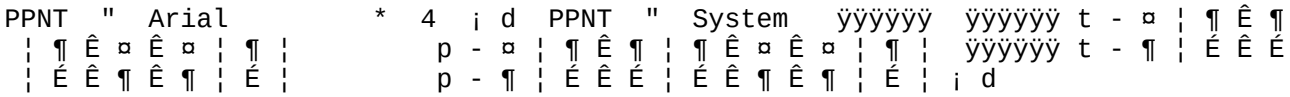
PPNT " Arial (-
3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - % 8 £ 8 8 £ % £ % 8
p - % 8 £ 8 8 £ % £ % 8 ¡ d

PPNT " Arial * 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - 8 J £ J
J £ 8 £ 8 J p - 8 J £ J J £ 8 £ 8 J ¡ d

PPNT " Arial * 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - J \ £ \
\ £ J £ J \ p - J \ £ \ \ £ J £ J \ ¡ d

PPNT " Arial * 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - € ¦ ’ Ê ’ ¦ ’ Ê € Ê € ¦ ’ ¦ p - € ¦ ’ Ê ’ ¦ ’ Ê € Ê € ¦ ’ ¦ ¡ d

PPNT " Arial (Œ µ 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ’ ¦ ¤ Ê ¤ ¦ ¤ Ê ’ Ê ’ ¦ ¤ ¦ p - ’ ¦ ¤ Ê ¤ ¦ ¤ Ê ’ Ê ’ ¦ ¤ ¦ ¡ d


PPNT " Arial
PPNT " System
t - ¤
t - ¶

PPNT " Arial *$ 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - € 9 ’] ’ 9 ’] €] € 9 ’ 9 p - € 9 ’] ’ 9 ’] €] € 9 ’ 9 ¡ d

PPNT " Arial (Œ H 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ’ 9 ¤] ¤ 9 ¤] ’] ’ 9 ¤ 9 p - ’ 9 ¤] ¤ 9 ¤] ’] ’ 9 ¤ 9 ¡ d

PPNT " Arial * 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¤ 9 ¶] ¶ 9 ¶] ¤] ¤ 9 ¶ 9 p - ¤ 9 ¶] ¶ 9 ¶] ¤] ¤ 9 ¶ 9 ¡ d

PPNT " Arial * 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¶ 9 É] É 9 É] ¶] ¶ 9 É 9 p - ¶ 9 É] É 9 É] ¶] ¶ 9 É 9 ¡ d

PPNT " Arial * 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - € ’ 7 ’
’ 7 € 7 € ’ p - € ’ 7 ’ ’ 7 € 7 € ’ ÿÿÿÿÿÿ t - ’ ¤ 7 ¤
¤ 7 ’ 7 ’ ¤ p - ’ ¤ 7 ¤ ¤ 7 ’ 7 ’ ¤ ¡ d

PPNT " Arial (ž " 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¤ ¶ 7
¶ ¶ 7 ¤ 7 ¤ ¶ p - ¤ ¶ 7 ¶ ¶ 7 ¤ 7 ¤ ¶ ¡ d

PPNT " Arial * 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¶ É 7 É
É 7 ¶ 7 ¶ É p - ¶ É 7 É É 7 ¶ 7 ¶ É ¡ d

PPNT " Arial * 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - € ’ £ ’ ’ £ € £ € ’ p - € ’ £ ’ ’ £ € £ € ’ ¡ d

PPNT " Arial (Œ 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ’ ¤ £ ¤ ¤ £ ’ £ ’ ¤ p - ’ ¤ £ ¤ ¤ £ ’ £ ’ ¤ ¡ d

PPNT " Arial * 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¤ ¶ £ ¶ ¶ £ ¤ £ ¤ ¶ p - ¤ ¶ £ ¶ ¶ £ ¤ £ ¤ ¶ ¡ d

PPNT " Arial * 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¶ É £ É
É £ ¶ £ ¶ É p - ¶ É £ É É £ ¶ £ ¶ É ¡ d

PPNT " Arial * 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - í ¦ ÿ Ê ÿ ¦ ÿ Ê í Ê í ¦ ÿ ¦ p - í ¦ ÿ Ê ÿ ¦ ÿ Ê í Ê í ¦ ÿ ¦ ¡ d

PPNT " Arial (ù µ 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ÿ ¦ Ê
¦ Ê ÿ Ê ÿ ¦ ¦ p - ÿ ¦ Ê ¦ Ê ÿ Ê ÿ ¦ ¦ ¡ d

PPNT " Arial * 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¦ # Ê # ¦ # Ê Ê ¦ # ¦ p - ¦ # Ê # ¦ # Ê Ê ¦ # ¦ ¡ d

PPNT " Arial * 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - # ¦ 5 Ê 5 ¦ 5 Ê # Ê # ¦ 5 ¦ p - # ¦ 5 Ê 5 ¦ 5 Ê # Ê # ¦ 5 ¦ ÿÿÿÿÿÿ t - í 9 ÿ] ÿ 9 ÿ] í] í 9 ÿ 9 p - í 9 ÿ] ÿ 9 ÿ] í] í 9 ÿ 9 ¡ d

PPNT " Arial (ù H 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ÿ 9]
9] ÿ] ÿ 9 9 p - ÿ 9] 9] ÿ] ÿ 9 9 ¡ d

PPNT " Arial * 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - 9 #] # 9 #]] 9 # 9 p - 9 #] # 9 #]] 9 # 9 ¡ d

PPNT " Arial * 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - # 9 5] 5 9 5] #] # 9 5 9 p - # 9 5] 5 9 5] #] # 9 5 9 ¡ d

PPNT " Arial * 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - í ÿ 7 ÿ
ÿ 7 í 7 í ÿ p - í ÿ 7 ÿ ÿ 7 í 7 í ÿ ¡ d

PPNT " Arial (ù " 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ÿ 7
7 ÿ 7 ÿ p - ÿ 7 7 ÿ 7 ÿ ÿÿÿÿÿÿ t - # 7
7 7 # p - # 7 # # 7 7 # ¡ d

PPNT " Arial *$ 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - # 5 7 5
5 7 # 7 # 5 p - # 5 7 5 5 7 # 7 # 5 ¡ d

PPNT " Arial * 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - í ÿ £ ÿ
ÿ £ í £ í ÿ p - í ÿ £ ÿ ÿ £ í £ í ÿ ¡ d

PPNT " Arial (ù 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ÿ £
£ ÿ £ ÿ p - ÿ £ £ ÿ £ ÿ ¡ d

PPNT " Arial * 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - # £ # # £ £ # p - # £ # # £ £ # ¡ d

PPNT " Arial * 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - # 5 £ 5
5 £ # £ # 5 p - # 5 £ 5 5 £ # £ # 5 ¡ d

PPNT " Arial * 4 ¡ d PPNT " System p f ˜ f ˜ ÿÿÿÿ
ÿÿ
ÿÿÿÿÿÿÿÿ t N — — — — ˜ ˜ ˜ ˜ ˜ ˜ ˜ — — —
— p Ó Ó î t N

p @ r @ [r t N p v v p q q q q q q q
q q q q q q p v p € f € ˜ € f € € ˜ t N } — ƒ € ƒ — ƒ —
, — , — ˜ ˜ ˜ € ˜ € ˜ ˜ ˜ ~ — ~ — ~ — } —
€ p € Ó € € Ó € î € t N } ƒ €
ƒ ƒ , , € € ~ ~ ~ } €
p € @ € r € @ € [€ r t N } p ƒ v € v ƒ p ƒ q , q , q q q q € q €
q q q ~ q ~ q ~ q } p € v p í f í ˜ í f í í ˜ t N ê — ï í ï — ï —
ï — î — î ˜ î ˜ í ˜ í ˜ ì ˜ ì ˜ ì ˜ ë — ë — ê — ê —
í p í Ó í í Ó í î í t N ê ï í
ï ï ï î î î í í ì ì ì ë ë ê ê í
p í @ í r í @ í [í r t N ê p ï v í v ï p ï q ï q î q î q î q í q í q ì
q ì q ì q ë q ë q ê q ê p í v ¡ d PPNT Century Schoolbook

(;

(a) ¡ d PPNT " System ¡ d PPNT Century Schoolbook *m (b) ¡ d PPNT " System ¡ d PPNT Century Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook + m (c) ¡ d PPNT " System p / ' / 5 / ' / 0 / 5 ÿÿÿÿÿÿ t N , 3 1 9 / 9 1 3 1 4 1 4 0 4 0 4 / 4 / 4 / 4 . 4 . 4 - 4 - 4 - 4 , 4 , 3 / 9 p - ' - 5 - ' - 0 - 5 t N « 3 ° 9 - 9 ° 3 ° 4 ¯ 4 ¯ 4 ¯ 4 ® 4 ® 4 - 4 - 4 - 4 ¬ 4 ¬ 4 « 4 « 4 « 3 - 9 p , ' , 5 , ' , 0 , 5 t N) 3 / 9 , 9 / 3 / 4 . 4 . 4 . 4 - 4 - 4 , 4 , 4 , 4 + 4 + 4 * 4 * 4) 3 , 9 ÿ" ! í ª !

í ª ü í ª ÿ

ÿÿÿÿ ª í - ÿÿÿÿÿÿ

í ª

t - © & Í & © & Í Í © & © p - © & Í & © & Í Í © & © ¡ d

PPNT " Arial ,

Helvetica

. +¸

1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - & © 8 Í 8 © 8 Í & Í & © 8 © p - & © 8 Í 8 © 8 Í & Í & © 8 © ¡ d

PPNT " Arial * 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - 8 © J Í J © J Í 8 Í 8 © J © p - 8 © J Í J © J Í 8 Í 8 © J © ¡ d

PPNT " Arial * 5 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - J © \ Í \ © \ Í J Í J © \ © p - J © \ Í \ © \ Í J Í J © \ © ¡ d

PPNT " Arial * 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - < & a & < & a a < & < p - < & a & < & a a < & < ¡ d

PPNT " Arial (
L 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - & < 8 a 8 < 8 a & a & < 8 <
p - & < 8 a 8 < 8 a & a & < 8 < ¡ d

PPNT " Arial * 5 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - 8 < J a J < J a 8 a 8 < J < p - 8 < J a J < J a 8 a 8 < J < ¡ d

PPNT " Arial * 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - J < \ a \ < \ a J a J < \ < p - J < \ a \ < \ a J a J < \ < ¡ d

PPNT " Arial * 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - & : &
& : : & p - & : & & : : & ¡ d

PPNT " Arial (
& 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - & 8 : 8 8 : & : & 8
p - & 8 : 8 8 : & : & 8 ¡ d

PPNT " Arial * 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - 8 J : J
J : 8 : 8 J p - 8 J : J J : 8 : 8 J ¡ d

PPNT " Arial * 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - J \ : \
\ : J : J \ p - J \ : \ \ : J : J \ ¡ d

PPNT " Arial * 5 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ƒ & § &
ƒ & § § ƒ & ƒ p - ƒ & § & ƒ & § § ƒ & ƒ ¡ d

PPNT " Arial (
’ 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - & ƒ 8 § 8 ƒ 8 § & § & ƒ 8 ƒ
p - & ƒ 8 § 8 ƒ 8 § & § & ƒ 8 ƒ ¡ d

PPNT " Arial * 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - 8 ƒ J § J
ƒ J § 8 § 8 ƒ J ƒ p - 8 ƒ J § J ƒ J § 8 § 8 ƒ J ƒ ¡ d

PPNT " Arial * 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - J ƒ \ § \
ƒ \ § J § J ƒ \ ƒ p - J ƒ \ § \ ƒ \ § J § J ƒ \ ƒ ¡ d

PPNT " Arial * 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - € © ’ Í ’ © ’ Í € Í € © ’ © p - € © ’ Í ’ © ’ Í € Í € © ’ © ¡ d

PPNT " Arial (Œ ¸ 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ’ © ¤ Í ¤ © ¤ Í ’ Í ’ © ¤ © p - ’ © ¤ Í ¤ © ¤ Í ’ Í ’ © ¤ © ¡ d

PPNT " Arial * 5 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¤ © ¶ Í ¶ © ¶ Í ¤ Í ¤ © ¶ © p - ¤ © ¶ Í ¶ © ¶ Í ¤ Í ¤ © ¶ © ¡ d

PPNT " Arial * 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¶ © È Í È © È Í ¶ Í ¶ © È © p - ¶ © È Í È © È Í ¶ Í ¶ © È © ¡ d

PPNT " Arial * 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - € < ’ a ’ < ’ a € a € < ’ < p - € < ’ a ’ < ’ a € a € < ’ < ¡ d

PPNT " Arial (Œ L 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ’ < ¤ a ¤ < ¤ a ’ a ’ < ¤ < p - ’ < ¤ a ¤ < ¤ a ’ a ’ < ¤ < ¡ d

PPNT " Arial * 5 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¤ < ¶ a ¶ < ¶ a ¤ a ¤ < ¶ < p - ¤ < ¶ a ¶ < ¶ a ¤ a ¤ < ¶ < ¡ d

PPNT " Arial * 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¶ < È a È < È a ¶ a ¶ < È < p - ¶ < È a È < È a ¶ a ¶ < È < ¡ d

PPNT " Arial * 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - € ’ : ’
’ : € : € ’ p - € ’ : ’ ’ : € : € ’ ¡ d

PPNT " Arial (Œ & 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ’ ¤ : ¤ ¤ : ’ : ’ ¤ p - ’ ¤ : ¤ ¤ : ’ : ’ ¤ ¡ d

PPNT " Arial * 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¤ ¶ : ¶
¶ : ¤ : ¤ ¶ p - ¤ ¶ : ¶ ¶ : ¤ : ¤ ¶ ¡ d

PPNT " Arial * 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¶ È : È
È : ¶ : ¶ È p - ¶ È : È È : ¶ : ¶ È ¡ d

PPNT " Arial * 5 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - € ƒ ’ § ’ ƒ ’ § € § € ƒ ’ ƒ p - € ƒ ’ § ’ ƒ ’ § € § € ƒ ’ ƒ ¡ d

PPNT " Arial (Œ ’ 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ’ ƒ ¤ §
¤ ƒ ¤ § ’ § ’ ƒ ¤ ƒ p - ’ ƒ ¤ § ¤ ƒ ¤ § ’ § ’ ƒ ¤ ƒ ¡ d

PPNT " Arial * 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¤ ƒ ¶ § ¶
ƒ ¶ § ¤ § ¤ ƒ ¶ ƒ p - ¤ ƒ ¶ § ¶ ƒ ¶ § ¤ § ¤ ƒ ¶ ƒ ¡ d

PPNT " Arial * 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¶ ƒ È § È
ƒ È § ¶ § ¶ ƒ È ƒ p - ¶ ƒ È § È ƒ È § ¶ § ¶ ƒ È ƒ ¡ d

PPNT " Arial * 4 ¡ d PPNT " System p j œ j … œ ÿÿÿÿ
ÿÿ
ÿÿÿÿÿÿÿÿ t N š š › › › › › › › › › › › ›
š ¡ d

PPNT Arial ¡ d PPNT " System ÿÿÿÿÿÿ
t -
‹
‹
‹
¡ d

PPNT Arial (€ 2s ¡ d PPNT " System p ×
× ò ÿÿÿÿÿÿ
ÿÿÿÿÿÿÿÿ t N

¡ d

PPNT Arial ¡ d PPNT " System ÿÿÿÿÿÿ
t -
ì ÷
ì ì ÷
÷
ì ¡ d

PPNT
Arial)m 2s ¡ d PPNT " System p C u C ^ u ÿÿÿ
ÿÿÿ
ÿÿÿÿÿÿÿÿ t N t z z t t t u u u u u u u u u t t
t z ¡ d

PPNT Arial ¡ d PPNT " System ÿÿÿÿÿÿ
t -
Y d
Y Y d
d
Y ¡ d

PPNT
Arial)m 1s ¡ d PPNT " System p € j € œ € j € … € œ ÿÿÿ
ÿÿÿ
ÿÿÿÿÿÿÿÿ t N } š ƒ € ƒ š ƒ › , › , › › › › € › € › › › › ~ › ~ ›
} š € ¡ d

PPNT Arial ¡ d PPNT " System ÿÿÿÿÿÿ
t - z † ‹ z † † ‹ z ‹ z ¡ d

PPNT Arial (ƒ € 1s ¡ d PPNT " System p € × €
€ × € ò € ÿÿÿÿÿÿ
ÿÿÿÿÿÿÿÿÿ t N } ƒ
€
ƒ ƒ , , € € ~ ~ } €
¡ d

PPNT Arial ¡ d PPNT " System ÿÿÿÿÿÿ
t - z ì † ÷ z ì † ì † ÷ z ÷ z ì ¡ d

PPNT
Arial)m 1s ¡ d PPNT " System p € C € u € C € ^ € u ÿÿÿ
ÿÿÿ
ÿÿÿÿÿÿÿÿ t N } t ƒ z € z ƒ t ƒ t , t , u u u u € u € u u u u ~ t ~ t
} t € z ¡ d

PPNT Arial ¡ d PPNT " System ÿÿÿÿÿÿ
t - z Y † d z Y † Y † d z d z Y ¡ d

PPNT Arial)m 1s ¡ d PPNT " System ¡ d PPNT Century Schoolbook

(; (a) ¡ d PPNT " System ¡ d PPNT Century Schoolbook *m (b) ¡ d PPNT " System p A * A 8 A * A 3 A 8 ÿÿÿÿÿ ÿÿÿÿÿÿÿÿ t N > 7 D < A < D 7 C 7 C 7 B 7 B 7 A 8 A 8 @ 8 @ 7 @ 7 ? 7 ? 7 > 7 > 7 A < p - * - 8 - * - 3 - 8 t N « 7 ° < - < ° 7 ° 7 ¯ 7 ¯ 7 ¯ 7 ® 7 ® 8 - 8 - 8 - 7 ¬ 7 ¬ 7 « 7 « 7 « 7 - < ÿÿÿÿÿ

t - \ < n a n < n a \ a \ < n < p - \ < n a n < n a \ a \ < n < ¡ d

PPNT " Arial

(h L 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - \ © n Í n © n Í \ Í \ © n © p - \ © n Í n © n Í \ Í \ © n © ¡ d

PPNT " Arial)l 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - \ n : n
n : \ : \ n p - \ n : n n : \ : \ n ¡ d

PPNT " Arial)n 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - \ ƒ n § n
ƒ n § \ § \ ƒ n ƒ p - \ ƒ n § n ƒ n § \ § \ ƒ n ƒ ¡ d

PPNT " Arial)l 5 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - È ƒ Û § Û
ƒ Û § È § È ƒ Û ƒ p - È ƒ Û § Û ƒ Û § È § È ƒ Û ƒ ¡ d

PPNT " Arial *l 5 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - È Û : Û
Û : È : È Û p - È Û : Û Û : È : È Û ¡ d

PPNT " Arial (Ô & 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - È © Û Í Û © Û Í È Í È © Û © p - È © Û Í Û © Û Í È Í È © Û © ¡ d

PPNT " Arial (Ô ¸ 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - È < Û a
Û < Û a È a È < Û < p - È < Û a Û < Û a È a È < Û < ¡ d

PPNT " Arial (Ô L 4 ¡ d PPNT " System p S — S ¥ S —
S S ¥ ÿÿÿÿÿÿ
ÿÿÿÿÿÿÿÿ t N P £ V © S © V £ U £ U ¤ U ¤ T ¤ T ¤ S ¤ S ¤ S ¤ R ¤ R ¤ Q ¤ Q ¤ Q £
P £ S © p e e e e
e t N b h e h g g g f f e e e d d c c c b
e p ¿ — ¿ ¥ ¿ —
¿ ¿ ¥ t N ½ £ Â © ¿ © Â £ Â £ Á ¤ Á ¤ Á ¤ À ¤ À ¤ ¿ ¤ ¿ ¤ ¿ ¤ ¾ ¤ ¾ ¤ ½ ¤ ½ £
½ £ ¿ © p Ñ Ñ Ñ Ñ
Ñ t N Ï Ô Ñ Ô Ô Ó Ó Ó Ò Ò Ñ Ñ Ñ Đ Đ Ï Ï Ï
Ñ p - ‰ a - s ‰ a ‰ s › s - s - a p - › Í ¿ ß › Í › ß - ß ¿ ß ¿ Í p - ¿ : Ñ L
¿ : ¿ L È L Ñ L Ñ : ÿ

Ž f € f p f ÿ

ÿÿÿÿ f Œ ¡ d PPNT Century Schoolbook -
f ,

Helvetica

. +

Let)″
and stop w)= hen with)? when ¡ d PPNT Century Schoolbook (

h), h)' h)œ h)/ h)! N ¡ d PPNT Century Schoolbook
(L s)& s)œ t)/ t ¡ d PPNT Century
Schoolbook (- 1)5 1)ñ 2 ¡ d PPNT Century Schoolbook

(

0 1)5 3) 1 ¡ d PPNT Symbol , Symbol (

' =)4 =) +)Ó ³ ¡ d PPNT Symbol
(O +)ñ + ¡ d PPNT Century Schoolbook

(

6 ,)R ,)Ù . ¡ d PPNT " System ÿ ¬ T l ž T l Ž
T l ÿ

ÿÿÿÿ l T Œ ¡ d PPNT Century Schoolbook -
T l ,

Helvetica

. +
h * h * h * h * h ¡ d PPNT Century Schoolbook
(

1 + 2 * 3 * 4 * 5 ¡ d PPNT Century Schoolbook

(
1 + 3) 1) 1) 4 (-
3) 4) 1) 13 (> 3) 13) 1) 40 (O 3) 40) 1) 121 ¡ d
PPNT Symbol , Symbol (
= + =) ´) +) = (-
=) ´) +) = (> =) ´) +) = (O =) ´) +) = ¡ d
PPNT Century Schoolbook (( )) (-
( )) (> ( )!) (O ( )!) ¡ d PPNT " System ÿ ú C
ì C Ü C ÿ

ÿÿÿÿ C Œ ¡ d PPNT Symbol -
C , Symbol

. + ë ¡ d PPNT Symbol) û ¡ d PPNT Century Schoolbook ,

Helvetica

(

```
  h )# h  ¡ d  PPNT     Century Schoolbook
  (     s )# s  ¡ d  PPNT     Symbol          (
- ¡ d  PPNT     Symbol

(
```

= ¡ d PPNT Century Schoolbook
(1 ¡ d PPNT Century Schoolbook

(

7 3 (

1 / ¡ d PPNT " System ÿï . ¶ ñ ¶ ñ ¶ ñ ÿ

ÿÿÿÿ ñ ¶ - ÿÿÿÿÿÿ

¶ ñ

t - 9 "] " 9 "]] 9 " 9 p - 9 "] " 9 "]] 9 " 9

¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial ,

Helvetica

. +I 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t -] " , "] " , ,] "] p -] " , "] " , ,] "] ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)$ 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - , " ¦ " , " ¦ ¦ , " , p - , " ¦ " , " ¦ ¦ , " , ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)$ 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¦ " Ê "
¦ " Ê Ê ¦ " ¦ p - ¦ " Ê " ¦ " Ê Ê ¦ " ¦ ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)$ 5 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - Ê " î "
Ê " î î Ê " Ê p - Ê " î " Ê " î î Ê " Ê ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)% 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - Y 9 k] k
9 k] Y] Y 9 k 9 p - Y 9 k] k 9 k] Y] Y 9 k 9 ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial (e I 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - Y] k , k] k , Y , Y] k] p - Y] k , k] k , Y , Y] k] ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)$ 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - Y , k ¦ k
, k ¦ Y ¦ Y , k , p - Y , k ¦ k , k ¦ Y ¦ Y , k , ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)$ 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - Y ¦ k Ê k
¦ k Ê Y Ê Y ¦ k ¦ p - Y ¦ k Ê k ¦ k Ê Y Ê Y ¦ k ¦ ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)$ 5 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - Y Ê k î k Ê k î Y î Y Ê k Ê p - Y Ê k î k Ê k î Y î Y Ê k Ê ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)% 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¡ 9 ³] ³ 9 ³] ¡] ¡ 9 ³ 9 p - ¡ 9 ³] ³ 9 ³] ¡] ¡ 9 ³ 9 ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial (- I 1 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¡] ³ , ³] ³ , ¡ , ¡] ³] p - ¡] ³ , ³] ³ , ¡ , ¡] ³] ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)$ 2 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¡ , ³ ¦ ³
, ³ ¦ ¡ ¦ ¡ , ³ , p - ¡ , ³ ¦ ³ , ³ ¦ ¡ ¦ ¡ , ³ , ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)$ 3 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¡ ¦ ³ Ê ³
¦ ³ Ê ¡ Ê ¡ ¦ ³ ¦ p - ¡ ¦ ³ Ê ³ ¦ ³ Ê ¡ Ê ¡ ¦ ³ ¦ ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)$ 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ¡ Ê ³ î ³ Ê ³ î ¡ î ¡ Ê ³ Ê p - ¡ Ê ³ î ³ Ê ³ î ¡ î ¡ Ê ³ Ê ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)% 5 ¡ d PPNT " System ¡ d PPNT Century Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook

(

(a) ¡ d PPNT " System ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook *H (b) ¡ d PPNT " System ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook + I (c) ¡ d PPNT " System p V o 9 y , o , q s € t t } t b t
` u ^ w ^ y] w] u \ t Z t Y t = t < s : q 9 o 9 ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p V o ¦ y î o î q î s í t ì t ê t Ï t Í u Ë w Ê y Ê w Ê u É t Ç t Å t ª t © s § q ¦ o ¦ ¡ d

PPNT Arial ¡ d PPNT " System ¡ d PPNT " System ¡ d PPNT 1
Courier New ¡ d PPNT " System ¡ d PPNT 1
Courier New ,
Courier

(

E Lb ¡ d PPNT " System ¡ d PPNT 1
Courier New ¡ d PPNT " System ¡ d PPNT 1
Courier New)‘ Ub ¡ d PPNT " System p & ″ + ″ + ″ ' ″ & ″ ÿÿÿÿÿÿ
ÿÿÿÿÿÿÿÿ t N " ‘ (— " ″ (— (– (–
(• ' • ' ″ ' ″ ' ″ ' ″ ' ″ ' ’ (’ (’ (‘ (‘ " ″ ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook (3 ˆ pivot ¡ d PPNT " System ¡ d PPNT 1
Courier New ¡ d PPNT " System ¡ d PPNT 1
Courier New (U E Lb ¡ d PPNT " System ¡ d PPNT 1
Courier New ¡ d PPNT " System ¡ d PPNT 1
Courier New)L M ¡ d PPNT " System ¡ d PPNT 1
Courier New ¡ d PPNT " System ¡ d PPNT 1
Courier
New)E Lb ¡ d PPNT " System ÿDU p ª S b ª S R ª S ÿ

ÿÿÿÿ S ª - ÿÿÿÿÿÿ

ª S

t - -) B) -) B B -) - p - -) B) -) B B -) -

ÿÿÿÿÿÿ t -) - ; B ; - ; B) B) - ; - p -) - ; B ; - ; B) B) - ; -

¡ d

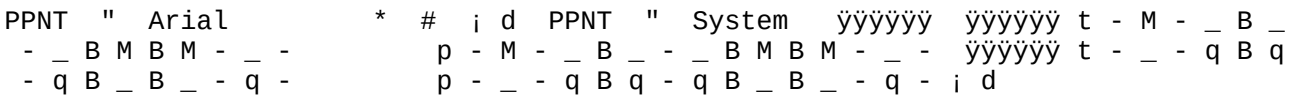
PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial ,

Helvetica

. +-
5 # ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ; - M B M - M B ; B ; - M -
p - ; - M B M - M B ; B ; - M - ¡ d

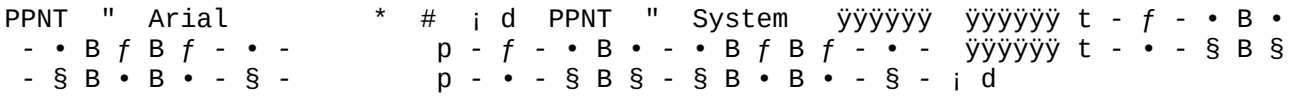
PPNT " Arial ¡ d PPNT " System ¡ d


PPNT " Arial * # ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - M - _ B _
- _ B M B M - _ - p - M - _ B _ - _ B M B M - _ - ÿÿÿÿÿÿ t - _ - q B q
- q B _ B _ - q - p - _ - q B q - q B _ B _ - q - ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial *$ # ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - q - ƒ B ƒ - ƒ B q B q - ƒ - p - q - ƒ B ƒ - ƒ B q B q - ƒ - ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d


PPNT " Arial
PPNT " System

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial *$ # ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - x) œ) x) œ œ x) x p - x) œ) x) œ œ x) x ¡ d

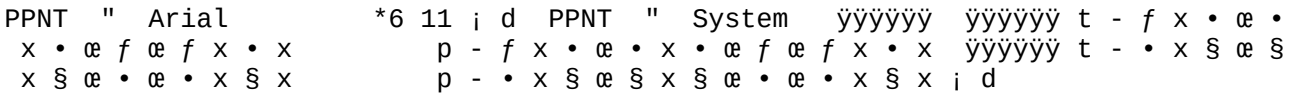
PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial (# ^ # ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t -) x ; œ ; x ; œ) œ) x ; x p -) x ; œ ; x ; œ) œ) x ; x ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial (5 … 16 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - M x _
œ _ x _ œ M œ M x _ x p - M x _ œ _ x _ œ M œ M x _ x ÿÿÿÿÿÿ t - _ x q
œ q x q œ _ œ _ x q x p - _ x q œ q x q œ _ œ _ x q x ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d


PPNT " Arial
x • œ ƒ œ ƒ x • x
x § œ • œ • x § x
*6 11 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ƒ x • œ •
p - ƒ x • œ • x • œ ƒ œ ƒ x • x ÿÿÿÿÿÿ t - • x § œ §
p - • x § œ § x § œ • œ • x § x ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial *6 22 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ƒ Ó • ÷ • Ó • ÷ ƒ ÷ ƒ Ó • Ó p - ƒ Ó • ÷ • Ó • ÷ ƒ ÷ ƒ Ó • Ó ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial (â # ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - • Ó § ÷ § Ó § ÷ • ÷ • Ó § Ó p - • Ó § ÷ § Ó § ÷ • ÷ • Ó § Ó ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial * 6 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - M Ó _ ÷ _
Ó _ ÷ M ÷ M Ó _ Ó p - M Ó _ ÷ _ Ó _ ÷ M ÷ M Ó _ Ó ÿÿÿÿÿÿ t - _ Ó q ÷ q
Ó q ÷ _ ÷ _ Ó q Ó p - _ Ó q ÷ q Ó q ÷ _ ÷ _ Ó q Ó ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial (k ß 27 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - M - _ Q _ - _ Q M Q M - _ - p - M - _ Q _ - _ Q M Q M - _ - ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial (Y < # ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - _ -
q Q q - q Q _ Q _ - q - p - _ - q Q q - q Q _ Q _ - q - ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial (k : 19 ¡ d PPNT " System p 0 t 0
t ÿÿÿÿÿÿ
ÿÿÿÿÿÿÿÿ t ¾ - . " 2 2 2 2 2 -
2 - 1 - 1 - 1 - 1 - 0 - 0 - 0 - / - / - / - / - . - . . . .
. ! . ! . ! . ! . " / " / " / " / " 0 " 0 " 0 " 1 " 1 ! 1 ! 1 ! 2 ! 2 2 2
2 t N s # x x # s " s " s ! s ! s ! s s s s s s - s - s s s
x ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p V 0 V t V 0 V t t ¾ T .
X 2 V 2 V 2 U 2 U 2 U 2 U 1 T 1 T 1 T 1 T 0 T 0 T 0 T / T / T / T / T . U . U .
U . U . V . V . V . W . W . W . X . X / X / X / X / X 0 X 0 X 0 X 1 X 1 X 1
W 1 W 2 W 2 W 2 V 2 V 2 t N S s Y x V x Y s X s X s X s W s W s V s V s V s U s
U s T s T s T s S s V x ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p Œ 0 Œ t Œ 0 Œ t t ¾ Š .
Ž 2 Œ 2 Œ 2 Œ 2 ‹ 2 ‹ 2 ‹ 1 ‹ 1 Š 1 Š 1 Š 0 Š 0 Š 0 Š / Š / Š / Š / ‹ . ‹ . ‹ .
‹ . Œ . Œ . Œ . Œ . . . . Ž . Ž . Ž / Ž / Ž / Ž / Ž 0 Ž 0 Ž 0 Ž 1 Ž 1 Ž 1
Ž 1 2 2 2 Œ 2 Œ 2 t N ‰ s x Œ x s s Ž s Ž s s s s Œ s Œ s ‹ s
‹ s ‹ s Š s Š s ‰ s Œ x ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p Œ Š Œ Î Œ Š Œ Î t ¾ Š ˆ
Ž Œ Œ Œ Œ Œ Œ Œ ‹ Œ ‹ Œ ‹ Œ ‹ Œ Š ‹ Š ‹ Š ‹ Š ‹ Š Š Š Š Š Š Š Š Š ‰ ‹ ‰ ‹ ‰ ‹ ‰
‹ ˆ Œ ˆ Œ ˆ Œ ˆ Œ ˆ ˆ ˆ ‰ Ž ‰ Ž ‰ Ž ‰ Ž Š Ž Š Ž Š Ž Š Ž ‹ Ž ‹ Ž ‹ Ž ‹ Ž Œ
Ž Œ Œ Œ Œ Œ Œ Œ Œ t N ‰ Í Ó Œ Ó Í Í Ž Í Ž Í Î Î Î Œ Î Œ Î ‹ Î
‹ Î ‹ Í Š Í Š Í ‰ Í Œ Ó ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p V Š V Î V Š V Î t ¾ T ˆ
X Œ V Œ V Œ U Œ U Œ U Œ U Œ T Œ T ‹ T ‹ T ‹ T ‹ T Š T Š T Š T Š T ‰ T ‰ U ‰ U ‰
U ˆ U ˆ V ˆ V ˆ V ˆ W ˆ W ˆ W ‰ W ‰ X ‰ X ‰ X Š X Š X Š X Š X ‹ X ‹ X ‹ X ‹ X Œ
W Œ W Œ W Œ W Œ V Œ V Œ t N S Í Y Ó V Ó Y Í X Í X Í X Í W Î W Î V Î V Î V Î U Î
U Î T Í T Í T Í S Í V Ó ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p V å V) V å V) t ¾ T ã X ç V ç V ç U ç U ç U ç U æ T æ T æ T æ T å T å T å T ä T ä T ä T ä T ã U ã U ã U ã U ã V ã V ã V ã W ã W ã W ã W ã X ã X ä X ä X ä X ä X å X å X å X æ X æ X æ W æ W ç W ç W ç V ç V ç t N S (Y - V -
Y (X (X (X (W (W (V (V (V (U (U (T (T (T (S (V - ¡ d

PPNT Arial ¡ d PPNT " System ¡ d PPNT " System ¡ d PPNT 1
Courier New ¡ d PPNT " System ¡ d PPNT 1
Courier New ,
Courier

(HashTable ¡ d PPNT " System ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial

(#
0 ¡ d PPNT " System ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial * 1 ¡ d PPNT " System ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial * 2 ¡ d PPNT " System ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial * 3 ¡ d PPNT " System ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial * 4 ¡ d PPNT " System ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial * 5 ¡ d PPNT " System ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial * 6 ¡ d PPNT " System ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT
Arial * 7 ¡ d PPNT " System ÿ z 5 l 5 \ 5
ÿ

ÿÿÿÿ 5 Œ -
5 "

ô # ¡ d PPNT Symbol , Symbol
. + (¡ d PPNT Symbol)!) ¡ d PPNT Century Schoolbook ,

Helvetica

(

5) 1) 2 ¡ d PPNT Symbol (- ¡ d PPNT Century Schoolbook) / ¡ d PPNT " System ÿú
X t 3
J t 3
: t 3 ÿ

ÿÿÿÿ 3 t - ÿÿÿÿÿÿ
t 3
t ~ Š ! © Š

Š
‹ “ – ™ £ ¦

© © © ¨ ¦ - £ ! ! ™ ! –
“ - ‹ Š Ś p ~ Š ! © Š

Š
‹ “ – ™ £ ¦

© © © ¨ ¦ - £ ! ! ™ ! – “ - ‹ Š Š ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial ,

Helvetica

. +” 20 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ & / E O 5 / 2 / / 1 , 2) 5 ' 8 & ; & ? & B ' F) I , K / M 2 N 5 O 9 N < M ? K B I D F E B E ? E ; D 8 B 5 ? 2 < 1 9 / 5 / p ~ & / E O 5 / 2 / / 1 , 2) 5 ' 8 & ; & ? & B ' F) I , K / M 2 N 5 O 9 N < M ? K B I D F E B E ? E ; D 8 B 5 ? 2 < 1 9 / 5 / ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT
Arial (8 < 7 ¡ d PPNT " System p N / ‹ ‹ / N ÿÿÿÿÿÿ ÿÿÿÿÿ
ÿ t ~ S \ r | c \ _] \ ^ Y ` V b T e S i S l S p T s V v Y x \ z _ | c | f | i
z l x o v q s r p r l r i q e o b l ` i ^ f] c \ p ~ S \ r | c \ _] \
^ Y ` V b T e S i S l S p T s V v Y x \ z _ | c | f | i z l x o v q s r p r l r
i q e o b l ` i ^ f] c \ ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT
Arial ++. 16 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ S r ! c _
\ Y V T
S

S S T V Y - \ _ ! c ! f ! i l - o q r r r

q
o l i f c p ~ S r ! c _ \ Y V T
S

S S T V Y - \ _ ! c ! f ! i l - o q r r r

q
o l i f c ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT
Arial (f 4 ¡ d PPNT " System p A W 4 A 4 W p A J W a W
a A J ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ & ä E 5 2 / ,) þ ' û & ø & ô & ñ ' í) ê ,
è / æ 2 å 5 ä 9 å < æ ? è B ê D í E ñ E ô E ø D û B þ ? < 9 5 p
~ & ä E 5 2 / ,) þ ' û & ø & ô & ñ ' í) ê , è / æ 2 å 5 ä 9 å < æ ?
è B ê D í E ñ E ô E ø D û B þ ? < 9 5 ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT
Arial (8 î 38 ¡ d PPNT " System p ¨ / å ¨ / å ÿÿÿÿÿÿ ÿÿÿ ÿÿÿ t ~ S r 1 c 1 _ 1 \ 0 Y . V + T (S % S ! S - T V Y \ _ c f i l o q r - r ! r % q (o + l . i 0 f 1 c 1 p ~ S r 1 c 1 _ 1 \ 0 Y . V + T (S % S ! S - T V Y \ _ c f i l o q r - r ! r % q (o + l . i 0 f 1 c 1 ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT
Arial +.. 43 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ S · r × c × _
Ö \ Õ Y Ó V Ñ T Î S Ê S Ç S Ã T À V ½ Y º \ ¹ _ · c · f · i ¹ l º o ½ q À r Ã r
Ç r Ê q Î o Ñ l Ó i Õ f Ö c × p ~ S · r × c × _ Ö \ Õ Y Ó V Ñ T Î S Ê S
Ç S Ã T À V ½ Y º \ ¹ _ · c · f · i ¹ l º o ½ q À r Ã r Ç r Ê q Î o Ñ l Ó i Õ f
Ö c × ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT
  Arial   (f Á 37 ¡ d PPNT " System p   A Ò W é A é W Ò p   A ÿ W
W   A ÿ ÿ
P   Æ 3
B   Æ 3
2   Æ 3   ÿ

ÿÿÿÿ 3 Æ - ÿÿÿÿÿÿ
Æ 3
t ~ ! !

-

! ! ! - - ! ! !

- p ~ ! !

-

! ! ! - - ! ! !

- ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial ,

Helvetica

. + 4 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ / = O -
/) / & 1 # 2 5 - 8 ; ? B - F I # K & M) N -
O 0 N 4 M 7 K 9 I ; F < B = ? < ; ; 8 9 5 7 2 4 1 0 / - / p ~ / = O -
/) / & 1 # 2 5 - 8 ; ? B - F I # K & M) N -
O 0 N 4 M 7 K 9 I ; F < B = ? < ; ; 8 9 5 7 2 4 1 0 / - / ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial +- 7 ¡ d PPNT " System p $ 1 $ 1 -
ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ 8 \ X | H \ D] A ^ > ` ; b : e 8 i 8 l 8 p : s ; v > x A z
D | H | K | O z R x T v V s W p X l W i V e T b R ` O ^ K] H \ p ~ 8 \
X | H \ D] A ^ > ` ; b : e 8 i 8 l 8 p : s ; v > x A z D | H | K | O z R x T v
V s W p X l W i V e T b R ` O ^ K] H \ ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT
Arial ++ 16 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ S Š s © c Š _ Š \ ‹ Y W U “ T –
S ™ T U W £ Y ¦ \ ¨ _ © c © f © j ¨ m ¦ o £ q r s ™ r –
q “ o m j ‹ f Š c Š p ~ S Š s © c Š _ Š \ ‹ Y W U “ T –
S ™ T U W £ Y ¦ \ ¨ _ © c © f © j ¨ m ¦ o £ q r s ™ r –
q “ o m j ‹ f Š c Š ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial +-
20 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ n · Ž × ~ · { · w ¹ t º r ½ p À
o Ã n Ç o Ê p Î r Ñ t Ó w Õ { Ö ~ × , Ö … Õ ˆ Ó Š Ñ Œ Î Ž Ê Ž Ç Ž Ã Œ À Š ½ ˆ º
… ¹ , · ~ · p ~ n · Ž × ~ · { · w ¹ t º r ½ p À o Ã n Ç o Ê p Î r Ñ t Ó
w Õ { Ö ~ × , Ö … Õ ˆ Ó Š Ñ Œ Î Ž Ê Ž Ç Ž Ã Œ À Š ½ ˆ º … ¹ , · ~ · ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial +-
37 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ ‰ ä © ™ ä –
å ’ æ è ê ‹ í Š ñ ‰ ô Š ø ‹ û þ ’ –
™ £ ¦ þ ¨ û © ø © ô © ñ ¨ í ¦ ê £ è æ å ™ ä p ~ ‰ ä ©
™ ä – å ’ æ è ê ‹ í Š ñ ‰ ô Š ø ‹ û þ ’ –
™ £ ¦ þ ¨ û © ø © ô © ñ ¨ í ¦ ê £ è æ å ™ ä ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial +-
38 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ ¥ Ä 1 ´ ± ® « ¨ ¦
¥ - ¥ ! ¥ % ¦ (¨ + « . ® 0 ± 1 ´ 1 ¸ 1 » 0 ¾ . Á + Ã (Ä % Ä ! Ä - Ã Á ¾
» ¸ ´ p ~ ¥ Ä 1 ´ ± ® « ¨ ¦ ¥ - ¥ ! ¥ % ¦ (¨ + « .
® 0 ± 1 ´ 1 ¸ 1 » 0 ¾ . Á + Ã (Ä % Ä ! Ä - Ã Á ¾ » ¸ ´ ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT
Arial +. 43 ¡ d PPNT " System p 5 L @ ^ @ ^ 5 L p P z [Œ [
Œ P z p k § v ¹ v ¹ k § p † Ô ‘ æ ‘ æ † Ô p ¢ ¬ ¬ ¢ ÿ
Ä ¦ 3
¶ ¦ 3
¦ ¦ 3 ÿ

ÿÿÿÿ 3 ¦ - ÿÿÿÿÿÿ
¦ 3
t ~ Š ! © Š

Š
‹ “ – ™ £ ¦

© © © ¨ ¦ - £ ! ! ™ ! –
“ - ‹ Š Ś p ~ Š ! © Š

Š
‹ “ – ™ £ ¦

© © © ¨ ¦ - £ ! ! ™ ! – “ - ‹ Š Š ¡ d

PPNT Arial ,

Helvetica

. +” 20 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ & / F O 6 / 2 / / 1 , 2) 5 ' 8 & ; & ? & B ' F) I , K / M 2 N 6 O 9 N < M @ K B I D F E B F ? E ; D 8 B 5 @ 2 < 1 9 / 6 / p ~ & / F O 6 / 2 / / 1 , 2) 5 ' 8 & ; & ? & B ' F) I , K / M 2 N 6 O 9 N < M @ K B I D F E B F ? E ; D 8 B 5 @ 2 < 1 9 / 6 / ¡ d

PPNT
Arial (9 < 7 ¡ d PPNT " System p N 0 ‹ ‹ 0 N ÿÿÿÿÿÿ ÿÿÿÿÿ
ÿ t ~ S \ s | c \ _] \ ^ Y ` V b U e S i S l S p U s V v Y x \ z _ | c | f | j
z m x o v q s r p s l r i q e o b m ` j ^ f] c \ p ~ S \ s | c \ _] \
^ Y ` V b U e S i S l S p U s V v Y x \ z _ | c | f | j z m x o v q s r p s l r
i q e o b m ` j ^ f] c \ ¡ d

PPNT Arial ++-
16 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ S s ! c _ \ Y V U
S

S S U V Y - \ _ ! c ! f ! j m - o q r s r

q
o m j f c p ~ S s ! c _ \ Y V U
S

S S U V Y - \ _ ! c ! f ! j m - o q r s r

q
o m j f c ¡ d

PPNT
Arial (f 4 ¡ d PPNT " System p A X 4 A 4 X p A J X a X
a A J ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ & ä F 6 2 / ,) þ ' û & ø & ô & ñ ' í) ê ,
è / æ 2 å 6 ä 9 å < æ @ è B ê D í E ñ F ô E ø D û B þ @ < 9 6 p
~ & ä F 6 2 / ,) þ ' û & ø & ô & ñ ' í) ê , è / æ 2 å 6 ä 9 å < æ @
è B ê D í E ñ F ô E ø D û B þ @ < 9 6 ¡ d

PPNT
Arial (9 î 38 ¡ d PPNT " System p ¨ 0 å ¨ 0 å ÿÿÿÿÿÿ ÿÿÿ ÿÿÿ t ~ S s 1 c 1 _ 1 \ 0 Y . V + U (S % S ! S - U V Y \ _ c f j m o q r - s ! r % q (o + m . j 0 f 1 c 1 p ~ S s 1 c 1 _ 1 \ 0 Y . V + U (S % S ! S - U V Y \ _ c f j m o q r - s ! r % q (o + m . j 0 f 1 c 1 ¡ d

PPNT Arial +.-
43 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ S · s × c × _ Ö \ Õ Y Ó V Ñ U Î S
Ê S Ç S Ã U À V ½ Y º \ ¹ _ · c · f · j ¹ m º o ½ q À r Ã s Ç r Ê q Î o Ñ m Ó j
Õ f Ö c × p ~ S · s × c × _ Ö \ Õ Y Ó V Ñ U Î S Ê S Ç S Ã U À V ½ Y º \
¹ _ · c · f · j ¹ m º o ½ q À r Ã s Ç r Ê q Î o Ñ m Ó j Õ f Ö c × ¡ d

PPNT
Arial (f Á 37 ¡ d PPNT " System p A Ò X é A é X Ò p A ÿ X
X A ÿ ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ ‰ © ‘ ‰ Š Š ‹ ‡ ” ’ • ™ ’ „ £
‡ ¦ Š § © ‘ © ” © ˜ § › ¦ £ Ÿ ™ • Ÿ ’ › ˜ ‹ ” Š ‘ ‰
p ~ ‰ © ‘ ‰ Š Š ‹ ‡ ” ’ • ™ ’ „ £ ‡ ¦ Š § © ‘ © ” © ˜ §
› ¦ £ Ÿ ™ • Ÿ ’ ” › ˜ ‹ ” Š ‘ ‰ ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial (” “ 18 ¡ d PPNT " System p o w … Ž … Ž o w ÿ
\ ¦ 3
N ¦ 3
> ¦ 3 ÿ

ÿÿÿÿ 3 ¦ - ÿÿÿÿÿÿ
¦ 3
t ~ Š ! © Š

Š
‹ “ – ™ £ ¦

© © © ¨ ¦ - £ ! ! ™ ! –
“ - ‹ Š Ś p ~ Š ! © Š

Š
‹ “ – ™ £ ¦

© © © ¨ ¦ - £ ! ! ™ ! – “ - ‹ Š Š ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial ,

Helvetica

. +” 37 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ & / F O 6 / 2 / / 1 , 2) 5 ' 8 & ; & ? & B ' F) I , K / M 2 N 6 O 9 N < M @ K B I D F E B F ? E ; D 8 B 5 @ 2 < 1 9 / 6 / p ~ & / F O 6 / 2 / / 1 , 2) 5 ' 8 & ; & ? & B ' F) I , K / M 2 N 6 O 9 N < M @ K B I D F E B F ? E ; D 8 B 5 @ 2 < 1 9 / 6 / ¡ d

PPNT
Arial (9 < 7 ¡ d PPNT " System p N 0 ‹ ‹ 0 N ÿÿÿÿÿÿ ÿÿÿÿÿ
ÿ t ~ S \ s | c \ _] \ ^ Y ` V b U e S i S l S p U s V v Y x \ z _ | c | f | j
z m x o v q s r p s l r i q e o b m ` j ^ f] c \ p ~ S \ s | c \ _] \
^ Y ` V b U e S i S l S p U s V v Y x \ z _ | c | f | j z m x o v q s r p s l r
i q e o b m ` j ^ f] c \ ¡ d

PPNT Arial ++-
16 ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ S s ! c _ \ Y V U
S

S S U V Y - \ _ ! c ! f ! j m - o q r s r

q
o m j f c p ~ S s ! c _ \ Y V U
S

S S U V Y - \ _ ! c ! f ! j m - o q r s r

q
o m j f c ¡ d

PPNT
Arial (f 4 ¡ d PPNT " System p A X 4 A 4 X p A J X a X
a A J ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ & ä F 6 2 / ,) þ ' û & ø & ô & ñ ' í) ê ,
è / æ 2 å 6 ä 9 å < æ @ è B ê D í E ñ F ô E ø D û B þ @ < 9 6 p
~ & ä F 6 2 / ,) þ ' û & ø & ô & ñ ' í) ê , è / æ 2 å 6 ä 9 å < æ @
è B ê D í E ñ F ô E ø D û B þ @ < 9 6 ¡ d

PPNT
Arial (9 î 38 ¡ d PPNT " System p ¨ 0 å ¨ 0 å ÿÿÿÿÿÿ ÿÿÿ ÿÿÿ t ~ S s 1 c 1 _ 1 \ 0 Y . V + U (S % S ! S - U V Y \ _ c f j m o q r - s ! r % q (o + m . j 0 f 1 c 1 p ~ S s 1 c 1 _ 1 \ 0 Y . V + U (S % S ! S - U V Y \ _ c f j m o q r - s ! r % q (o + m . j 0 f 1 c 1 ¡ d

PPNT Arial +.-
43 ¡ d PPNT " System p A ÿ X X A ÿ ÿÿÿÿÿÿ ÿÿÿÿÿÿ t ~ ‰ © ‘ ‰
Š Š ‹ ‡ ” ’ • ™ ’ ” £ ‡ ¦ Š § © ‘ © ” © ˜ § › ¦ £ Ÿ
™ • Ÿ ’ › ˜ ‹ ” Š ‘ ‰ p ~ ‰ © ‘ ‰ Š Š ‹ ‡ ” ’ • ™ ’ ” £ ‡ ¦ Š § © ‘ © ” © ˜ § › ¦ £ Ÿ ™ • Ÿ ’ › ˜ ‹ ” Š ‘ ‰ ¡ d

PPNT
Arial (” “ 18 ¡ d PPNT " System p o w … Ž … Ž o w ÿ Ö
Å 0 È Å 0 ¸ Å 0 ÿ

ÿÿÿÿ 0 Å - ÿÿÿÿÿÿ
Å 0
t †

ª 0 Î - ª ª ¬ ® º

o

¾

Â

Å È Ë Í Î - Î " Î % Í) Ë + È . Å / Â 0 ¾ 0 º / · . ³ + °) ® % ¬ " ª - ª p †

ª 0 Î - ª ª ¬ ® º

o

¾

Â

Å È Ë Í Î - Î " Î % Í) Ë + È . Å / Â 0 ¾ 0 º / · . ³ + °) ® % ¬ " ª - ª ¡ d PPNT Century Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook ,

Helvetica

. +°! black ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t † T b x † f b c b _ c \ e Y h W k U n T r T v U y W } Y € \ , _ „ c … f † j … n „ q , t € v } w y x v x r w n v k t h q e n c j b f b p † T b x † f b c b _ c \ e Y h W k U n T r T v U y W } Y € \ , _ „ c … f † j … n „ q , t € v } w y x v x r w n v k t h q e n c j b f b ¡ d PPNT Century Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook (i l red ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t † T ò x f ò c ó _ ô \ ö Y ø W û U ÿ T T U W

Y \ _ c f j n q t v

w
x x w ÿ v û t ø q ö n ô j ó f ò p † T ò x f ò c ó _ ô \ ö Y ø W û
U ÿ T T U
W

Y \ _ c f j n q t v

w
x x w ÿ v û t ø q ö n ô j ó f ò ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook)‘ red ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t † Á = ¯
« § ¤ ¡ Ÿ " ž &) -
ž 1 Ÿ 4 ¡ 8 ¤ : § < « = ¯ = ³ = ¶ < º : ¼ 8 ¿ 4 À 1 Á - Á) À & ¿ " ¼ -
º ¶ ³ ¯ p † Á = ¯ « § ¤ ¡ Ÿ " ž &) -
ž 1 Ÿ 4 ¡ 8 ¤ : § < « = ¯ = ³ = ¶ < º : ¼ 8 ¿ 4 À 1 Á - Á) À & ¿ " ¼ -
º ¶ ³ ¯ ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p + Y ¯ + ¯ Y p s 8
¢ g s g ¢ 8 p + É Y ø + É Y ø ÿÿÿÿÿÿ t ~ Ÿ ¿ ; ¯ « ¨ ¥ -
¢ ! ¡ $ Ÿ (Ÿ + Ÿ / ¡ 2 ¢ 5 ¥ 8 ¨ : « ; ¯ ; ² ; ¶ : ¹ 8 » 5 ½ 2 ¾ / ¿ + ¾ (½ $
» ! ¹ ¶ ² ¯ p ~ Ÿ ¿ ; ¯ « ¨ ¥ -
¢ ! ¡ $ Ÿ (Ÿ + Ÿ / ¡ 2 ¢ 5 ¥ 8 ¨ : « ; ¯ ; ² ; ¶ : ¹ 8 » 5 ½ 2 ¾ / ¿ + ¾ (½ $
» ! ¹ ¶ ² ¯ ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook (² $ red ¡ d PPNT " System ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook

(B ¡ d PPNT " System ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook (W T A ¡ d PPNT " System ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook (

X ¡ d PPNT " System ¡ d PPNT Century Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook (W C ¡ d PPNT " System ¡ d PPNT Century Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook

(† e parent ¡ d PPNT " System ¡ d PPNT Century Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook)“ uncle ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t †

ª 2 Î ª ª ¬ ® º ³ .

o

¾ Â Å È Ë Í Î
Î $ Î (Í + Ë . È 0 Å 1 Â 2 ¾ 2 º 1 · 0 ³ . ° + ® (¬ $ ª ª p †

ª 2 Î ª ª ¬ ® º ³ .

o

¾ Â Å È Ë Í Î
Î $ Î (Í + Ë . È 0 Å 1 Â 2 ¾ 2 º 1 · 0 ³ . ° + ® (¬ $ ª
ª ¡ d PPNT Century Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook (# ´ red ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t † W b { † i b e b a c ^ e [h Y k X n W r W v X y Y } [€ ^ , a „ e … i † m … p „ s , v € x } z y { v { r z n x k v h s e p c m b i b p † W b { † i b e b a c ^ e [h Y k X n W r W v X y Y } [€ ^ , a „ e … i † m … p „ s , v € x } z y { v { r z n x k v h s e p c m b i b ¡ d PPNT Century Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook (l g black ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t † W ò { i ò e ó a ô ^ ö [ø Y û X ÿ W W X Y

[^ a e i m p s v x

z
{ { z ÿ x û v ø s ö p ô m ó i ò p † W ò { i ò e ó a ô ^ ö [ø Y û
X ÿ W W X
Y

[^ a e i m p s v x

z
{ { z ÿ x û v ø s ö p ô m ó i ò ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook)‘ black ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t † Ÿ Ã =
± ® ª § ¤ ¢ " & Ÿ) Ÿ -
1 ¢ 4 ¤ 8 § : ª < ® = ± = µ = ¹ < ¼ : ¿ 8 Á 4 Ã 1 Ã - Ã) Ã & Á " ¿ -
¼ ¹ µ ± p † Ÿ Ã = ± ® ª § ¤ ¢ " & Ÿ) Ÿ -
1 ¢ 4 ¤ 8 § : ª < ® = ± = µ = ¹ < ¼ : ¿ 8 Á 4 Ã 1 Ã - Ã) Ã & Á " ¿ -
¼ ¹ µ ± ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook (´ $ red ¡ d PPNT " System p - \ ¯ -
¯ \ p v 8 ¥ g v g ¥ 8 p - É \ ø - É \ ø ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook

(B ¡ d PPNT " System ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook (Z T A ¡ d PPNT " System ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook (¢

X ¡ d PPNT " System ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook (Z C ¡ d PPNT " System p T ¼ ò ¼ T ¼ ¦ ¼ ò ¼ ÿÿÿÿÿÿ
ÿÿÿÿÿÿÿÿÿ t N ð ¹ ÷ À ÷ ¼ ð ¹ ñ ¹ ñ º ñ º ñ » ñ » ñ ¼ ñ ¼ ñ ½ ñ ½ ñ ¾ ñ ¾ ñ ¿ ñ ¿
ð À ÷ ¼ ÿ8b Ü y ’ Ü y , Ü y ÿ

ÿÿÿÿ y Ü - ÿÿÿÿÿÿ
Ü y
t †

ª 0 Ï - ª « ¬ ® °

»

¾

Â

Å É Ë Í Î - Ï " Î % Í (Ë + É - Å / Â 0 ¾ 0 » / · -
³ + ° (® % ¬ " « - ª p †

ª 0 Ï - ª « ¬ ® °

»

¾

Â

Å É Ë Í Î - Ï " Î % Í (Ë + É - Å / Â 0 ¾ 0 » / · -
³ + ° (® % ¬ " « - ª ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook ,

Helvetica

. +°! black ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t † T b x † f b b b _ c \ e Y h W k U n T r T v U y W } Y € \ ƒ _ … b † f † j † m … q ƒ t € v } w y x v x r w n v k t h q e m c j b f b p † T b x † f b b b _ c \ e Y h W k U n T r T v U y W } Y € \ ƒ _ … b † f † j † m … q ƒ t € v } w y x v x r w n v k t h q e m c j b f b ¡ d PPNT Century Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook (i l red ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t † T ó x f ó b ó _ ô \ ö Y ù W ü U ÿ T T U W

Y \ _ b f j m q t v

w
x x w ÿ v ü t ù q ö m ô j ó f ó p † T ó x f ó b ó _ ô \ ö Y ù W ü
U ÿ T T U
W

Y \ _ b f j m q t v

w
x x w ÿ v ü t ù q ö m ô j ó f ó ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook)Œ black ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t † Á =
¯ « § ¤ ¡ Ÿ " & * -
1 Ÿ 4 ¡ 8 ¤ : § < « = ¯ = ² = ¶ < ¹ : ¼ 8 ¾ 4 À 1 Á - Á * À & ¾ " ¼ -
¹ ¶ ² ¯ p † Á = ¯ « § ¤ ¡ Ÿ " & * -
1 Ÿ 4 ¡ 8 ¤ : § < « = ¯ = ² = ¶ < ¹ : ¼ 8 ¾ 4 À 1 Á - Á * À & ¾ " ¼ -
¹ ¶ ² ¯ ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p * Y ° * ° Y p s 8
¢ g s g ¢ 8 p * É Y ø * É Y ø ÿÿÿÿÿÿ t ~ Ÿ ¿ ; « ¨ ¥ -
¢ ! $ Ÿ (Ÿ + Ÿ / 2 ¢ 5 ¥ 8 ¨ : « ; ¯ ; ² ; µ : ¸ 8 » 5 ½ 2 ¾ / ¿ + ¾ (½ $
» ! ¸ µ ² p ~ Ÿ ¿ ; « ¨ ¥ -
¢ ! $ Ÿ (Ÿ + Ÿ / 2 ¢ 5 ¥ 8 ¨ : « ; ¯ ; ² ; µ : ¸ 8 » 5 ½ 2 ¾ / ¿ + ¾ (½ $
» ! ¸ µ ² ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook (² $ red ¡ d PPNT " System ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook

(B ¡ d PPNT " System ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook (W U A ¡ d PPNT " System ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook (

X ¡ d PPNT " System ¡ d PPNT Century Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook (W C ¡ d PPNT " System ¡ d PPNT Century Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook

(… e parent ¡ d PPNT " System ¡ d PPNT Century Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook)” uncle ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t †

ª 2 Ï ª « ¬ ® ° ³ .

»

¾ Â Å É Ë Í Î Ï # Î ' Í * Ë - É / Å 1 Â 2 ¾ 2 » 1 · / ³ -
° * ® ' ¬ # « ª p †

ª 2 Ï ª « ¬ ® ° ³ .

»

¾ Â Å É Ë Í Î Ï # Î ' Í * Ë - É / Å 1 Â 2 ¾ 2 » 1 · / ³ -
° * ® ' ¬ # « ª ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook (# ° black ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t † V
b z † h b d b a c] e [h X k W n V r V v W y X } [€] ƒ a … d † h † l † p … s
ƒ v € x } y y z v z r y n x k v h s e p c l b h b p † V b z † h b d b a
c] e [h X k W n V r V v W y X } [€] ƒ a … d † h † l † p … s ƒ v € x } y y z
v z r y n x k v h s e p c l b h b ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook (k l red ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t † V ó z
h ó d ó a ô] ö [ù X ü W ÿ V V W
X

[ ] a d h l p s v x

y
z z y ÿ x ü v ù s ö p ô l ó h ó p † V ó z h ó d ó a ô] ö [ù X ü
W ÿ V V W
X

[ ] a d h l p s v x

y
z z y ÿ x ü v ù s ö p ô l ó h ó ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook)‘ red ¡ d PPNT " System p - [° - ° [p -
É [ø - É [ø ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook

(A ¡ d PPNT " System ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook (Y U X ¡ d PPNT " System ¡ d PPNT Century
Schoolbook ¡ d PPNT " System ¡ d PPNT Century
Schoolbook (e C ¡ d PPNT " System p T ¼ ò ¼ T ¼ ¦ ¼ ò ¼ ÿÿÿÿÿÿ
ÿÿÿÿÿÿÿÿ t N ð ¹ ÷ À ÷ ¼ ð ¹ ð ¹ ð º ñ º ñ » ñ » ñ ¼ ñ ¼ ñ ½ ñ ½ ñ ¾ ñ ¿ ð ¿ ð ¿
ð À ÷ ¼ p s s p » 8 Ê G » 8 Ê G p s s
p s ê ø s ø ê p » Ê » -
Ê ¡ d PPNT Symbol ¡ d PPNT " System ¡ d PPNT Symbol ,
Symbol

(‰ â d ¡ d PPNT " System ¡ d PPNT Symbol ¡ d PPNT " System ¡ d PPNT Symbol (‰ ‘ g ¡ d PPNT " System ¡ d PPNT Symbol ¡ d PPNT " System ¡ d PPNT Symbol (Ò H b ¡ d PPNT " System ¡ d PPNT Symbol ¡ d PPNT " System ¡ d PPNT Symbol (Ò a ¡ d PPNT " System ¡ d PPNT Symbol ¡ d PPNT " System ¡ d PPNT Symbol (‰ " e ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ

t † Ÿ ; Ã ` ± ; - < © = ¦ ? £ A ¡ D Ÿ H Ÿ L Ÿ O Ÿ S ¡ W £ Z ¦ \ © ^ - _ ± ` ´ _ ¸ ^ » \ ¾ Z À W Â S Ã O Ã L Â H À D ¾ A » ? ¸ = ´ < ± ; p † Ÿ ; Ã ` ± ; - < © = ¦ ? £ A ¡ D Ÿ H Ÿ L Ÿ O Ÿ S ¡ W £ Z ¦ \ © ^ - _ ± ` ´ _ ¸ ^ » \ ¾ Z À W Â S Ã O Ã L Â H À D ¾ A » ? ¸ = ´ < ± ; ¡ d PPNT Century Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook

(´ A black ¡ d PPNT " System p u ¤ A u ¤ A ¡ d PPNT Century Schoolbook ¡ d PPNT " System ¡ d PPNT Century Schoolbook

(U B ¡ d PPNT " System p u X ƒ g u g ƒ X ¡ d PPNT Symbol ¡ d PPNT " System ¡ d PPNT Symbol

(Œ P a ¡ d PPNT " System p u ƒ u ƒ ¡ d PPNT Symbol ¡ d PP NT " System ¡ d PPNT Symbol)@ b ¡ d PPNT " System p u ê ƒ ø u ø ƒ ê ¡ d PPNT Symbol ¡ d PPNT " System ¡ d PPNT Symbol)S g ¡ d PPNT " System p ½ Z Ì i ½ Z Ì i p ½ 2 Ì A ½ A Ì 2 ¡ d PPNT Symbol ¡ d PPNT " System ¡ d PPNT Symbol +HH d ¡ d PPNT " System ¡ d PP NT Symbol ¡ d PPNT " System ¡ d PPNT Symbol)@ e ¡ d PPNT " S ystem ÿ [€ ž Ó Zø ž ÓZè ® ÿ

ÿÿÿÿ ® - ÿÿÿÿÿÿ

®

t - 2 2 2 p - 2 2 2

¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial ,

Helvetica

. +"

¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - 2 2 2

p - 2 2 2 ÿÿÿÿÿÿ t - M i M i i M M

p - M i M i i M M ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)2 abe ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - M i
M i i M M p - M i M i i M M ÿÿÿÿÿÿ t - „ Ÿ
„ Ÿ Ÿ „ „ p - „ Ÿ „ Ÿ Ÿ „ „ ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)9 art ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - „ Ÿ
„ Ÿ Ÿ „ „ p - „ Ÿ „ Ÿ Ÿ „ „ ÿÿÿÿÿÿ t - º Õ
º Õ Õ º º p - º Õ º Õ Õ º º ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)4 ben ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - º Õ
º Õ Õ º º p - º Õ º Õ Õ º º ÿÿÿÿÿÿ t - ð

ð

ð ð p - ð

ð

ð ð ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)7 bob ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ð

ð

ð ð p - ð

ð

ð ð ÿÿÿÿÿÿ t - ' B ' B B ' ' p - ' B ' B B
' ' ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)8 cal ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ' B
' B B ' ' p - ' B ' B B ' ' ÿÿÿÿÿÿ t -] x
] x x]] p -] x] x x]] ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)5 cat ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t -] x
] x x]] p -] x] x x]] ÿÿÿÿÿÿ t - ″
″ ″ p - ″ ″ ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)6 dan ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ″ ¯
″ ¯ ¯ ″ ″ p - ″ ¯ ″ ¯ ¯ ″ ″ ÿÿÿÿÿÿ t - Ê å
Ê å å Ê Ê p - Ê å Ê å å Ê Ê ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

```
PPNT  "  Arial         )6 don  ¡ d  PPNT  "  System   ÿÿÿÿÿÿ  ÿÿÿÿÿÿ t -   Ê   å
   Ê   å   å   Ê   Ê         p -   Ê   å   Ê   å   å   Ê   Ê p     %   I   %   I

ÿÿÿÿÿÿÿÿ t ¾   "   '   '   '   '   &   &   &   &   &   %   %   %   %   $   $   $
   #   #   #   #   #   #   "   "   "   #   #   #   #   #   #   $   $   $   %   %
   %   %   &   &   &   &   &   '   '   ' t N   H   M   M   H   H   H   H   H   H
   I   I   I   H   H   H   H   H   H   M ¡ d
```

PPNT

Arial	¡	d	PPNT	"	System		¡	d	PPNT	"	System		p		[		[		t	¾	Y
]	]	]	]	]	]	\	\	\	\	[	[	[	[	Z	Z	Z	Z	Y	Y		
Y	Y	Y	Y	Y	Y	Y	Y	Y	Z	Z	Z	Z	[	[	[	[	\	\	\		
\	]	]	]	]	]	t	N	~		„	„	~	~	~							
		~	~	~	„	¡	d														

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p ‘ ¶ ‘ ¶ t ¾
“ “ “ “ “ “ “ “ ’ ’ ’ ’ ’ ’ ’
’ ’ ’ ’ ’ ’ ’ “
“ “ “ “ “ “ t N ´ º º ´ µ µ µ µ µ µ µ µ µ
µ µ µ µ ´ º ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p È ì È ì t ¾ Æ
Ê Ê Ê Ê Ê É É É É É È È È Ç Ç Ç Ç Æ Æ Æ
Æ Æ Æ Æ Æ Æ Æ Æ Æ Æ Ç Ç Ç Ç È È È É É É
É É Ê Ê Ê Ê t N ë ð ð ë ë ë ë ë ë ì ì ì ë
ë ë ë ë ë ð ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p þ # þ # t ¾ ü
ÿ ÿ ÿ ÿ þ þ þ þ ý ý ý ý ü
ü ü ü ü ü ü ü ü ý ý ý ý þ þ þ þ ÿ ÿ ÿ ÿ
t N ! ' ' ! ! " " " " " " " "
" " " ! ! ' ¡ d

PPNT

Arial	¡	d	PPNT	"	System	¡	d	PPNT	"	System	p		5	Y	5	Y	t	¾	2
7	7	7	7	6	6	6	6	6	5	5	5	5	4	4	4	3	3	3	3
3	3	2	2	2	3	3	3	3	3	3	4	4	4	5	5	5	5	6	6
6	6	6	7	7	7	t	N	X	]	]	X	X	X	X	X	X	X	X	X
X	X	X	X	X	]	¡	d												

PPNT

Arial	¡	d	PPNT	"	System		¡	d	PPNT	"	System		p		k		k	t	¾	i
m	m	m	m	m	m	l	l	l	l	k	k	k	k	j	j	j	j	i	i
i	i	i	i	i	i	i	i	i	j	j	j	j	k	k	k	k	l	l	l
l	m	m	m	m	m	t	N	Ž	”	”	Ž	Ž	Ž						
		Ž	Ž	Ž	”	¡	d												

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p ¡ Æ ¡ Æ t ¾ Ÿ
£ £ £ £ £ £ £ £ ¢ ¢ ¢ ¢ ¡ ¡ ¡ Ÿ
Ÿ Ÿ Ÿ Ÿ Ÿ Ÿ Ÿ Ÿ ¡ ¡ ¡ ¢ ¢ ¢ ¢ £
£ £ £ £ £ £ t N Ä Ê Ê Ä Å Å Å Å Å Å Å Å Å
Å Å Å Å Ä Ê ¡ d

PPNT

Arial	¡	d	PPNT	"	System		¡	d	PPNT	"	System		p		Ø	ü	Ø	ü	t ¾	Õ
Ú	Ú	Ú	Ú	Ù	Ù	Ù	Ù	Ù	Ø	Ø	Ø	Ø	×	×	×	Ö	Ö	Ö	Ö	
Ö	Ö	Õ	Õ	Õ	Ö	Ö	Ö	Ö	Ö	Ö	×	×	×	Ø	Ø	Ø	Ø	Ù	Ù	
Ù	Ù	Ù	Ú	Ú	Ú	t	N	û			û	û	û	û	û	û	ü	ü	ü	û
û	û	û	û	û		¡	d													

PPNT Arial ¡ d PPNT " System ¡ d PPNT " System ÿÿÿÿÿÿ
t - 8 F 2 F F 2 8 2 8 F p - 8 F 2 F F 2 8 2 8 F
¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial (A " # ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - F S 2
S S 2 F 2 F S p - F S 2 S S 2 F 2 F S ÿÿÿÿÿÿ t - 8 M F i
F M F i 8 i 8 M F M p - 8 M F i F M F i 8 i 8 M F M ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)2 abe ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - F M S i
S M S i F i F M S M p - F M S i S M S i F i F M S M ÿÿÿÿÿÿ t - 8 „ F Ÿ
F „ F Ÿ 8 Ÿ 8 „ F „ p - 8 „ F Ÿ F „ F Ÿ 8 Ÿ 8 „ F „ ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)9 art ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - F „ S Ÿ
S „ S Ÿ F Ÿ F „ S „ p - F „ S Ÿ S „ S Ÿ F Ÿ F „ S „ ÿÿÿÿÿÿ t - 8 º F Õ
F º F Õ 8 Õ 8 º F º p - 8 º F Õ F º F Õ 8 Õ 8 º F º ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)4 ben ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - F º S Õ
S º S Õ F Õ F º S º p - F º S Õ S º S Õ F Õ F º S º ÿÿÿÿÿÿ t - 8 ð F

F ð F

8 ð F ð p - 8 ð F

F ð F

8 ð F ð ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)7 bob ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - F ð S

S ð S

F

F ð S ð p - F ð S

S ð S

F

F ð S ð ÿÿÿÿÿÿ t - 8 ' F B F ' F B 8 B 8 ' F ' p - 8 ' F B F ' F B 8 B 8 ' F ' ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

```
PPNT  "  Arial         )8 cal  ¡ d  PPNT  "  System   ÿÿÿÿÿÿ  ÿÿÿÿÿÿ t - F ' S B
 S ' S B F B F ' S '          p - F ' S B S ' S B F B F ' S '  ÿÿÿÿÿÿ t - 8 ] F x
 F ] F x 8 x 8 ] F ]          p - 8 ] F x F ] F x 8 x 8 ] F ] ¡ d
```

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)5 cat ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - F] S x
S] S x F x F] S] p - F] S x S] S x F x F] S] ÿÿÿÿÿÿ t - 8 ″ F ¯
F ″ F ¯ 8 ¯ 8 ″ F ″ p - 8 ″ F ¯ F ″ F ¯ 8 ¯ 8 ″ F ″ ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)6 dan ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - F ″ S ¯
S ″ S ¯ F ¯ F ″ S ″ p - F ″ S ¯ S ″ S ¯ F ¯ F ″ S ″ ÿÿÿÿÿÿ t - 8 Ê F å
F Ê F å 8 å 8 Ê F Ê p - 8 Ê F å F Ê F å 8 å 8 Ê F Ê ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)6 don ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - F Ê S å
S Ê S å F å F Ê S Ê p - F Ê S å S Ê S å F å F Ê S Ê p L % L I L % L I

ÿÿÿÿÿÿÿÿ t ¾ J " N ' L ' L ' L ' K & K & K & K & K & J % J % J % J % J $ J $ J $
K # K # K # K # K # L # L " L " M " M # M # M # N # N # N # N $ N $ N $ N % N %
N % N % N & N & N & M & M & M ' M ' L ' t N I H O M L M O H O H N H N H M H M H
M I L I L I K H K H K H J H J H I H L M ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p L [L L [L t ¾ J Y
N] L] L] L] K] K] K \ K \ K \ J \ J [J [J [J [J Z J Z K Z K Z K Y K Y
K Y L Y L Y L Y M Y M Y M Y M Y N Y N Z N Z N Z N Z N [N [N [N \ N \ N \
N \ M] M] M] M] L] t N I ~ O „ L „ O ~ O ~ N ~ N M M M L L K
K K J ~ J ~ I ~ L „ ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p L ‘ L ¶ L ‘ L ¶ t ¾ J
N “ L “ L “ L “ K “ K “ K “ K “ K ’ J ’ J ’ J ’ J ‘ J ‘ J ‘ J K K K K
K L L L M M M M N N N N N ‘ N ‘ N ‘ N ’ N ’ N ’ N ’ N “
N “ M “ M “ M “ M “ L “ t N I ´ O º L º O ´ O µ N µ N µ M µ M µ M µ L µ L µ K µ
K µ K µ J µ J µ I ´ L º ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p L È L ì L È L ì t ¾ J Æ
N Ê L Ê L Ê L Ê K Ê K É K É K É K É J É J È J È J È J Ç J Ç J Ç K Ç K Æ K Æ K Æ
K Æ L Æ L Æ L Æ M Æ M Æ M Æ M Æ N Æ N Æ N Ç N Ç N Ç N Ç N È N È N È N É N É N É
N É M É M Ê M Ê M Ê L Ê t N I ë O ð L ð O ë O ë N ë N ë M ë M ë M ì L ì L ì K ë
K ë K ë J ë J ë I ë L ð ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p L þ L # L þ L # t ¾ J ü
N L L L K K K K ÿ K ÿ J ÿ J ÿ J þ J þ J þ J þ J ý K ý K ý K ý K ü
K ü L ü L ü L ü M ü M ü M ü M ü N ý N ý N ý N ý N þ N þ N þ N þ N ÿ N ÿ N ÿ N ÿ
N M M M M L t N I ! O ' L ' O ! O ! N " N " M " M " M " L " L " K "
K " K " J " J ! I ! L ' ¡ d

PPNT

Arial ¡ d PPNT " System ¡ d PPNT " System p L 5 L Y L 5 L Y t ¾ J 2
N 7 L 7 L 7 L 7 K 6 K 6 K 6 K 6 K 6 J 5 J 5 J 5 J 5 J 4 J 4 J 4 K 3 K 3 K 3 K 3
K 3 L 3 L 2 L 2 M 2 M 3 M 3 M 3 N 3 N 3 N 3 N 3 N 4 N 4 N 4 N 5 N 5 N 5 N 5 N 6 N 6
N 6 M 6 M 6 M 7 M 7 L 7 t N I X O] L] O X O X N X N X M X M X M X L X L X K X
K X K X J X J X I X L] ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p L k L L k L t ¾ J i
N m L m L m L m K m K m K l K l K l J l J k J k J k J k J j J j K j K j K i K i
K i L i L i L i M i M i M i M i N i N j N j N j N j N k N k N k N k N l N l N l
N l M m M m M m M m L m t N I Ž O ″ L ″ O Ž O Ž N Ž N M M M L L K
K K J Ž J Ž I Ž L ″ ¡ d

PPNT
  Arial ¡ d PPNT " System ¡ d PPNT " System p L ¡ L Æ L ¡ L Æ t ¾ J Ÿ
N £ L £ L £ L £ K £ K £ K £ K £ K ¢ J ¢ J ¢ J ¢ J ¡ J ¡ J ¡ J K K K K Ÿ
K Ÿ L Ÿ L Ÿ L Ÿ M Ÿ M Ÿ M Ÿ M Ÿ N N N N N ¡ N ¡ N ¡ N ¢ N ¢ N ¢ N ¢ N £
N £ M £ M £ M £ M £ L £ t N I Ä O Ê L Ê O Ä O Å N Å N Å M Å M Å M Å L Å L Å K Å
K Å K Å J Å J Å I Ä L Ê ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p L Ø L ü L Ø L ü t ¾ J Õ
N Ú L Ú L Ú L Ú K Ù K Ù K Ù K Ù J Ø J Ø J Ø J Ø J × J × J × K Ö K Ö K Ö K Ö
K Ö L Ö L Õ L Õ M Õ M Ö M Ö M Ö N Ö N Ö N Ö N × N × N × N Ø N Ø N Ø N Ø N Ù N Ù
N Ù M Ù M Ù M Ú M Ú L Ú t N I û O L O û O û N û N û M û M û M ü L ü L ü K û
K û K û J û J û I û L ¡ d

PPNT Arial ¡ d PPNT " System ¡ d PPNT " System ÿÿÿÿÿ
t - S a 2 a a 2 S 2 S a p - S a 2 a a 2 S 2 S a
ÿÿÿÿÿ t - S M a i a M a i S i S M a M p - S M a i a M a i S i S M a M
ÿÿÿÿÿ t - S º a Õ a º a Õ S Õ S º a º p - S º a Õ a º a Õ S Õ S º a º
ÿÿÿÿÿ t - S ' a B a ' a B S B S ' a ' p - S ' a B a ' a B S B S ' a '
ÿÿÿÿÿ t - S ″ a ¯ a ″ a ¯ S ¯ S ″ a ″ p - S ″ a ¯ a ″ a ¯ S ¯ S ″ a ″
p Z % Z I Z % Z I
ÿÿÿÿÿÿÿÿ t ¾ X " \ ' Z ' Z ' Y ' Y & Y & Y & X & X & X % X % X % X % X $ X $ X $
X # X # Y # Y # Y # Y # Z " Z " Z " [# [# [# [# \ # \ # \ $ \ $ \ $ \ % \ %
\ % \ % \ & \ & [& [& [& [' Z ' Z ' t N W H] M Z M] H \ H \ H \ H [H [H
Z I Z I Z I Y H Y H X H X H W H W H Z M ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p Z [Z ¶ Z [Z ¶ t ¾ X Y
\ ] Z] Z] Y] Y] Y] Y \ X \ X \ X \ X [X [X [X [X Z X Z X Z X Z Y Y Y Y
Y Y Y Y Z Y Z Y Z Y [Y [Y [Y [Y \ Z \ Z \ Z \ Z \ [\ [\ [\ [\ \ \ \ \ \
[\ [ ] [ ] [ ] Z] Z] t N W ´] º Z º] ´ \ µ \ µ \ µ [µ [µ Z µ Z µ Z µ Y µ
Y µ X µ X µ W µ W ´ Z º ¡ d

PPNT
  Arial ¡ d PPNT " System ¡ d PPNT " System p   Z È Z # Z È Z # t ¾ X Æ
\ Ê Z Ê Z Ê Y Ê Y Ê Y É Y É X É X É X É X È X È X È X Ç X Ç X Ç X Ç X Æ Y Æ Y Æ
Y Æ Y Æ Z Æ Z Æ Z Æ [Æ [Æ [Æ [Æ \ Æ \ Ç \ Ç \ Ç \ Ç \ È \ È \ È \ É \ É \ É
[É [É [Ê [Ê Z Ê Z Ê t N W !] ' Z '] ! \ ! \ " \ " [" [" Z " Z " Z " Y "
Y " X " X " W ! W ! Z ' ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p Z 5 Z Z 5 Z t ¾ X 2
\ 7 Z 7 Z 7 Y 7 Y 6 Y 6 Y 6 X 6 X 6 X 5 X 5 X 5 X 5 X 4 X 4 X 4 X 3 X 3 Y 3 Y 3
Y 3 Y 3 Z 2 Z 2 Z 2 [3 [3 [3 [3 \ 3 \ 3 \ 4 \ 4 \ 4 \ 5 \ 5 \ 5 \ 5 \ 6 \ 6
[6 [6 [6 [7 Z 7 Z 7 t N W Ž] ″ Z ″] Ž \ Ž \ Ž \ [[Z Z Z Y
Y X X Ž W Ž W Ž Z ″ ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p Z ¡ Z ü Z ¡ Z ü t ¾ X Ÿ
\ £ Z £ Z £ Y £ Y £ Y £ Y £ X £ X ¢ X ¢ X ¢ X ¢ X ¡ X ¡ X ¡ X X X Y Y Ÿ
Y Ÿ Y Ÿ Z Ÿ Z Ÿ Z Ÿ [Ÿ [Ÿ [Ÿ [\ \ \ \ ¡ \ ¡ \ ¡ \ ¢ \ ¢ \ ¢ \ ¢ \ £
[£ [£ [£ [£ Z £ Z £ t N W û] Z] û \ û \ û \ û [û [û Z ü Z ü Z ü Y û
Y û X û X û W û W û Z ¡ d

PPNT Arial ¡ d PPNT " System ¡ d PPNT " System ÿÿÿÿÿÿ
t - u ƒ 2 ƒ ƒ 2 u 2 u ƒ p - u ƒ 2 ƒ ƒ 2 u 2 u ƒ
¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial (~ " # ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ƒ 2
2 ƒ 2 ƒ p - ƒ 2 2 ƒ 2 ƒ ÿÿÿÿÿÿ t - u M ƒ i
ƒ M ƒ i u i u M ƒ M p - u M ƒ i ƒ M ƒ i u i u M ƒ M ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)2 abe ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ƒ M i
M i ƒ i ƒ M M p - ƒ M i M i ƒ i ƒ M M ÿÿÿÿÿÿ t - u „ ƒ Ÿ
ƒ „ ƒ Ÿ u Ÿ u „ ƒ „ p - u „ ƒ Ÿ ƒ „ ƒ Ÿ u Ÿ u „ ƒ „ ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)9 art ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ƒ „ Ÿ
„ Ÿ ƒ Ÿ ƒ „ „ p - ƒ „ Ÿ „ Ÿ ƒ Ÿ ƒ „ „ ÿÿÿÿÿÿ t - u º ƒ Õ
ƒ º ƒ Õ u Õ u º ƒ º p - u º ƒ Õ ƒ º ƒ Õ u Õ u º ƒ º ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)4 ben ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ƒ º Õ
º Õ ƒ Õ ƒ º º p - ƒ º Õ º Õ ƒ Õ ƒ º º ÿÿÿÿÿÿ t - u ð ƒ

f ð f

u

u ð *f* ð p - u ð *f*

f ð f

u

u ð ƒ ð ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)7 bob ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ƒ ð

ð

f

f ð ð p - *f* ð

ð

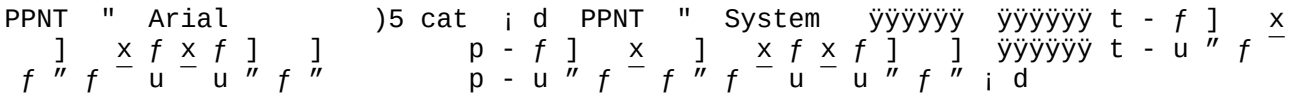
f

ƒ ð ð ÿÿÿÿÿÿ t - u ' ƒ B ƒ ' ƒ B u B u ' ƒ ' p - u ' ƒ B ƒ ' ƒ B u B u ' ƒ ' ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)8 cal ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - *ƒ* ' B
' B *ƒ* B *ƒ* ' ' p - *ƒ* ' B ' B *ƒ* B *ƒ* ' ' ÿÿÿÿÿÿ t - u] *ƒ* x
ƒ] *ƒ* x u x u] *ƒ*] p - u] *ƒ* x *ƒ*] *ƒ* x u x u] *ƒ*] ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d


PPNT " Arial)5 cat ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ƒ] x
] x ƒ x ƒ]] p - ƒ] x] x ƒ x ƒ]] ÿÿÿÿÿÿ t - u ″ ƒ
ƒ ″ ƒ u u ″ ƒ ″ p - u ″ ƒ ƒ ″ ƒ u u ″ ƒ ″ ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)6 dan ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ƒ ″ ¯
″ ¯ ƒ ¯ ƒ ″ ″ p - ƒ ″ ¯ ″ ¯ ƒ ¯ ƒ ″ ″ ÿÿÿÿÿÿ t - u Ê ƒ å
ƒ Ê ƒ å u å u Ê ƒ Ê p - u Ê ƒ å ƒ Ê ƒ å u å u Ê ƒ Ê ¡ d

PPNT " Arial ¡ d PPNT " System ¡ d

PPNT " Arial)6 don ¡ d PPNT " System ÿÿÿÿÿÿ ÿÿÿÿÿÿ t - ƒ Ê å
Ê å ƒ å ƒ Ê Ê p - ƒ Ê å Ê å ƒ å ƒ Ê Ê p ‰ % ‰ I ‰ % ‰ I

ÿÿÿÿÿÿÿÿ t ¾ ‡ " ‹ ' ‰ ' ‰ ' ‰ & ˆ & ˆ & ˆ & ˆ % ‡ % ‡ % ‡ % ‡ $ ‡ $ ˆ $
ˆ # ˆ # ˆ # ˆ # ‰ # ‰ # ‰ " ‰ " Š " Š # Š # ‹ # ‹ # ‹ # ‹ # ‹ $ ‹ $ ‹ $ ‹ % ‹ %
‹ % ‹ % ‹ & ‹ & ‹ & ‹ & Š & Š ' Š ' ‰ ' t N ‡ H Œ M ‰ M Œ H Œ H ‹ H ‹ H ‹ H Š H
Š I ‰ I ‰ I ‰ H ˆ H ˆ H ‡ H ‡ H ‡ H ‰ M ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p ‰ [‰ ‰ [‰ t ¾ ‡ Y
‹] ‰] ‰] ‰] ‰] ˆ] ˆ \ ˆ \ ˆ \ ˆ \ ‡ [‡ [‡ [‡ [‡ Z ˆ Z ˆ Z ˆ Z ˆ Y ˆ Y
‰ Y ‰ Y ‰ Y ‰ Y Š Y Š Y Š Y ‹ Y ‹ Y ‹ Z ‹ Z ‹ Z ‹ Z ‹ [‹ [‹ [‹ [‹ \ ‹ \ ‹ \
‹ \ ‹] Š] Š] Š] ‰] t N ‡ ~ Œ „ ‰ „ Œ ~ Œ ~ ‹ ~ ‹ ‹ Š Š ‰ ‰ ‰
ˆ ˆ ‡ ~ ‡ ~ ‡ ~ ‰ „ ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p ‰ ‘ ‰ ¶ ‰ ‘ ‰ ¶ t ¾ ‡
‹ “ ‰ “ ‰ “ ‰ “ ‰ “ ˆ “ ˆ “ ˆ “ ˆ ’ ˆ ’ ‡ ’ ‡ ’ ‡ ‘ ‡ ‘ ‡ ‘ ˆ ˆ ˆ ˆ ˆ
‰ ‰ ‰ ‰ Š Š Š ‹ ‹ ‹ ‹ ‹ ‹ ‘ ‹ ‘ ‹ ‘ ‹ ’ ‹ ’ ‹ ’ ‹ ’ ‹ “
‹ “ ‹ “ Š “ Š “ Š “ ‰ “ t N ‡ ´ Œ º ‰ º Œ ´ Œ µ ‹ µ ‹ µ ‹ µ Š µ Š µ ‰ µ ‰ µ ‰ µ
ˆ µ ˆ µ ‡ µ ‡ µ ‡ ´ ‰ º ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p ‰ È ‰ ì ‰ È ‰ ì t ¾ ‡ Æ
‹ Ê ‰ Ê ‰ Ê ‰ Ê ‰ Ê ˆ É ˆ É ˆ É ˆ É ‡ È ‡ È ‡ È ‡ Ç ‡ Ç ˆ Ç ˆ Ç ˆ Æ ˆ Æ ˆ Æ
‰ Æ ‰ Æ ‰ Æ ‰ Æ Š Æ Š Æ Š Æ ‹ Æ ‹ Æ ‹ Æ ‹ Ç ‹ Ç ‹ Ç ‹ Ç ‹ È ‹ È ‹ È ‹ É ‹ É ‹ É
‹ É ‹ É Š Ê Š Ê Š Ê ‰ Ê t N ‡ ë Œ ð ‰ ð Œ ë Œ ë ‹ ë ‹ ë ‹ ë Š ë Š ì ‰ ì ‰ ì ‰ ë
ˆ ë ˆ ë ‡ ë ‡ ë ‡ ë ‰ ð ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p ‰ þ ‰ # ‰ þ ‰ # t ¾ ‡ ü
‹ ‰ ‰ ‰ ‰ ˆ ˆ ˆ ÿ ˆ ÿ ˆ ÿ ‡ ÿ ‡ þ ‡ þ ‡ þ ‡ þ ˆ ý ˆ ý ˆ ý ˆ ý ˆ ü
‰ ü ‰ ü ‰ ü ‰ ü Š ü Š ü Š ü ‹ ü ‹ ý ‹ ý ‹ ý ‹ ý ‹ þ ‹ þ ‹ þ ‹ þ ‹ ÿ ‹ ÿ ‹ ÿ ‹ ÿ
‹ ‹ Š Š Š ‰ t N ‡ ! Œ ' ‰ ' Œ ! Œ ! ‹ " ‹ " ‹ " Š " Š " ‰ " ‰ " ‰ "
ˆ " ˆ " ‡ " ‡ ! ‡ ! ‰ ' ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p ‰ 5 ‰ Y ‰ 5 ‰ Y t ¾ ‡ 2
‹ 7 ‰ 7 ‰ 7 ‰ 7 ‰ 6 ˆ 6 ˆ 6 ˆ 6 ˆ 6 ˆ 5 ‡ 5 ‡ 5 ‡ 5 ‡ 4 ‡ 4 ˆ 4 ˆ 3 ˆ 3 ˆ 3 ˆ 3
‰ 3 ‰ 3 ‰ 2 ‰ 2 Š 2 Š 3 Š 3 ‹ 3 ‹ 3 ‹ 3 ‹ 3 ‹ 4 ‹ 4 ‹ 4 ‹ 5 ‹ 5 ‹ 5 ‹ 5 ‹ 6 ‹ 6
‹ 6 ‹ 6 Š 6 Š 7 Š 7 ‰ 7 t N ‡ X Œ] ‰] Œ X Œ X ‹ X ‹ X ‹ X Š X Š X ‰ X ‰ X ‰ X
ˆ X ˆ X ‡ X ‡ X ‡ X ‰] ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p ‰ k ‰ ‰ k ‰ t ¾ ‡ i
‹ m ‰ m ‰ m ‰ m ‰ m ˆ m ˆ l ˆ l ˆ l ‡ k ‡ k ‡ k ‡ k ‡ j ˆ j ˆ j ˆ j ˆ i ˆ i
‰ i ‰ i ‰ i ‰ i Š i Š i Š i ‹ i ‹ i ‹ j ‹ j ‹ j ‹ j ‹ k ‹ k ‹ k ‹ k ‹ l ‹ l ‹ l
‹ l ‹ m Š m Š m Š m ‰ m t N ‡ Ž Œ ” ‰ ” Œ Ž Œ Ž ‹ Ž ‹ ‹ Š Š ‰ ‰ ‰
ˆ ˆ ‡ Ž ‡ Ž ‡ Ž ‰ ” ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p ‰ ¡ ‰ Æ ‰ ¡ ‰ Æ t ¾ ‡ Ÿ
‹ £ ‰ £ ‰ £ ‰ £ ‰ £ ˆ £ ˆ £ ˆ £ ˆ ¢ ˆ ¢ ‡ ¢ ‡ ¢ ‡ ¡ ‡ ¡ ‡ ¡ ˆ ˆ ˆ ˆ ˆ Ÿ
‰ Ÿ ‰ Ÿ ‰ Ÿ ‰ Ÿ Š Ÿ Š Ÿ Š Ÿ ‹ Ÿ ‹ ‹ ‹ ‹ ‹ ¡ ‹ ¡ ‹ ¡ ‹ ¢ ‹ ¢ ‹ ¢ ‹ ¢ ‹ £
‹ £ ‹ £ Š £ Š £ Š £ ‰ £ t N ‡ Ä Œ Ê ‰ Ê Œ Ä Œ Å ‹ Å ‹ Å ‹ Å Š Å Š Å ‰ Å ‰ Å ‰ Å
ˆ Å ˆ Å ‡ Å ‡ Å ‡ Ä ‰ Ê ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p ‰ Ø ‰ ü ‰ Ø ‰ ü t ¾ ‡ Õ
‹ Ú ‰ Ú ‰ Ú ‰ Ú ‰ Ù ˆ Ù ˆ Ù ˆ Ù ˆ Ø ‡ Ø ‡ Ø ‡ Ø ‡ × ‡ × ˆ × ˆ Ö ˆ Ö ˆ Ö
‰ Ö ‰ Ö ‰ Õ ‰ Õ Š Õ Š Ö Š Ö ‹ Ö ‹ Ö ‹ Ö ‹ Ö ‹ × ‹ × ‹ × ‹ Ø ‹ Ø ‹ Ø ‹ Ø ‹ Ù ‹ Ù
‹ Ù ‹ Ù Š Ù Š Ú Š Ú ‰ Ú t N ‡ û Œ ‰ Œ û Œ û ‹ û ‹ û ‹ û Š û Š ü ‰ ü ‰ ü ‰ û
ˆ û ˆ û ‡ û ‡ û ‡ û ‰ ¡ d

PPNT Arial ¡ d PPNT " System ¡ d PPNT " System ÿÿÿÿÿÿ

t - ž 2 ž ž 2 2 ž p - ž 2 ž ž 2 2 ž

ÿÿÿÿÿÿ t - M ž i ž M ž i i M ž M p - M ž i ž M ž i i M ž M

ÿÿÿÿÿÿ t - º ž Õ ž º ž Õ Õ º ž º p - º ž Õ ž º ž Õ Õ º ž º

ÿÿÿÿÿÿ t - ' ž B ž ' ž B B ' ž ' p - ' ž B ž ' ž B B ' ž '

ÿÿÿÿÿÿ t - ″ ž ¯ ž ″ ž ¯ ¯ ″ ž ″ p - ″ ž ¯ ž ″ ž ¯ ¯ ″ ž ″

p — % — I — % — I

ÿÿÿÿÿÿÿÿ t ¾ • " ™ ' – ' – ' - ' - & - & -

& • & • & • % • % • % • % • $ • $ • $ • # • # - # - # - # - # — " — " — " —

~ # ~ # ~ # ~ # ™ # ™ $ ™ $ ™ $ ™ % ™ % ™ % ™ & ~ & ~ & ~ & ~ & — ' — ' —

' t N ″ H š M — M š H ™ H ™ H ™ H ~ H ~ H — I — I - I - H - H • H • H • H ″ H —

M ¡ d

PPNT Arial ¡ d PPNT " System ¡ d PPNT " System p — [— ¶ — [—
¶ t ¾ • Y ™] —] —] –] –] –] –
\ • \ • \ • \ • [• [• [• [• Z • Z • Z • Z – Y – Y – Y – Y — Y — Y — Y —
Y ˜ Y ˜ Y ˜ Y ˜ Z ™ Z ™ Z ™ Z ™ [™ [™ [™ [™ \ ™ \ ˜ \ ˜ \ ˜] ˜] —] —] —
] t N ″ ´ š º — º š ´ ™ µ ™ µ ™ µ ˜ µ ˜ µ — µ — µ – µ – µ – µ • µ • µ • µ ″ ´ —
º ¡ d

PPNT Arial ¡ d PPNT " System ¡ d PPNT " System p — È — # — È —
t ¾ • Æ ™ Ê — Ê — Ê – Ê – É –
É • É • É • É • È • È • È • Ç • Ç • Ç • Ç • Æ – Æ – Æ – Æ – Æ — Æ — Æ — Æ —
Æ ˜ Æ ˜ Æ ˜ Æ ˜ Æ ™ Ç ™ Ç ™ Ç ™ Ç ™ È ™ È ™ È ™ É ™ É ˜ É ˜ É ˜ É ˜ Ê — Ê — Ê —
Ê t N ″ ! š ' — ' š ! ™ ! ™ " ™ " ˜ " ˜ " _ " _ " _ " _ " _ " • " • " • ! ″ ! —
' ¡ d

PPNT Arial ¡ d PPNT " System ¡ d PPNT " System p — 5 — — 5 —
t ¾ • 2 ™ 7 — 7 — 7 – 7 – 6 – 6 –
6 • 6 • 6 • 5 • 5 • 5 • 5 • 4 • 4 • 4 • 3 • 3 – 3 – 3 – 3 – 3 — 2 — 2 — 2 —
3 ˜ 3 ˜ 3 ˜ 3 ˜ 3 ™ 3 ™ 4 ™ 4 ™ 4 ™ 5 ™ 5 ™ 5 ™ 5 ™ 6 ˜ 6 ˜ 6 ˜ 6 ˜ 6 — 7 — 7 —
7 t N ″ Ž š ″ — ″ š Ž ™ Ž ™ Ž ™ ~ ~ — — – – – • • Ž • Ž ″ Ž —
″ ¡ d

PPNT Arial ¡ d PPNT " System ¡ d PPNT " System p — ¡ — ü — ¡ —
ü t ¾ • Ÿ ™ £ — £ — £ – £ – £ –
£ • £ • ¢ • ¢ • ¢ • ¢ • ¡ • ¡ • ¡ • • • – – Ÿ – Ÿ – Ÿ — Ÿ — Ÿ — Ÿ —
Ÿ ˜ Ÿ ˜ Ÿ ˜ ˜ ™ ™ ™ ¡ ™ ¡ ™ ¡ ™ ¢ ™ ¢ ™ ¢ ™ ¢ ˜ £ ˜ £ ˜ £ ˜ £ — £ — £ —
£ t N ″ û š — š û ™ û ™ û ™ û ˜ û ˜ û — ü — ü – ü – û – û • û • û • û ″ û —
¡ d

PPNT Arial ¡ d PPNT " System ¡ d PPNT " System ÿÿÿÿÿÿ
t - ž « 2 « « 2 ž 2 ž « p - ž « 2 « « 2 ž 2 ž «
ÿÿÿÿÿÿ t - ž M « i « M « i ž i ž M « M p - ž M « i « M « i ž i ž M « M
ÿÿÿÿÿÿ t - ž ' « B « ' « B ž B ž ' « ' p - ž ' « B « ' « B ž B ž ' « '
p ¥ % ¥ I ¥ % ¥ I
ÿÿÿÿÿÿÿÿ t ¾ £ " § ' ¥ ' ¤ ' ¤ ' ¤ & £ & £ & £ & £ & £ % £ % £ % £ % £ $ £ $ £ $
£ # £ # £ # £ # ¤ # ¤ # ¤ " ¥ " ¥ " ¥ # ¥ # ¦ # ¦ # ¦ # ¦ # $ § $ § $ § % § %
§ % ¦ % ¦ & ¦ & ¦ & ¦ & ¥ & ¥ ' ¥ ' ¥ ' t N ¢ H § M ¥ M § H § H § H ¦ H ¦ H ¥ H
¥ I ¥ I ¤ I ¤ H £ H £ H £ H ¢ H ¢ H ¥ M ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p ¥ [¥ # ¥ [¥ # t ¾ £ Y
§] ¥] ¤] ¤] ¤] £] £ \ £ \ £ \ £ \ £ [£ [£ [£ [£ Z £ Z £ Z £ Z £ Y £ Y
¤ Y ¤ Y ¤ Y ¥ Y ¥ Y ¥ Y ¥ Y ¦ Y ¦ Y ¦ Z ¦ Z § Z § [§ [§ [§ [¦ \ ¦ \ ¦ \
¦ \ ¦] ¥] ¥] ¥] ¥] t N ¢ ! § ' ¥ ' § ! § ! § " ¦ " ¦ " ¥ " ¥ " ¥ " ¤ " ¤ "
£ " £ " £ " ¢ ! ¢ ! ¥ ' ¡ d

PPNT
Arial ¡ d PPNT " System ¡ d PPNT " System p ¥ 5 ¥ ü ¥ 5 ¥ ü t ¾ £ 2
§ 7 ¥ 7 ¤ 7 ¤ 7 ¤ 6 £ 6 £ 6 £ 6 £ 6 £ 5 £ 5 £ 5 £ 5 £ 4 £ 4 £ 4 £ 3 £ 3 £ 3 £ 3
¤ 3 ¤ 3 ¤ 2 ¥ 2 ¥ 2 ¥ 3 ¥ 3 ¦ 3 ¦ 3 ¦ 3 ¦ 3 ¦ 4 § 4 § 4 § 5 § 5 § 5 ¦ 5 ¦ 6 ¦ 6
¦ 6 ¦ 6 ¥ 6 ¥ 7 ¥ 7 ¥ 7 t N ¢ û § ¥ § û § û § û ¦ û ¦ û ¥ û ¥ ü ¥ ü ¤ ü ¤ û
£ û £ û £ û ¢ û ¢ û ¥ ¡ d

PPNT Arial ¡ d PPNT " System ¡ d PPNT " System ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial (
0 ¡ d PPNT " System ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial *7 0 ¡ d PPNT " System ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial *
1 ¡ d PPNT " System ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial *0 0 ¡ d PPNT " System ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial *
1 ¡ d PPNT " System ¡ d

PPNT Arial ¡ d PPNT " System ¡ d

PPNT Arial *

2 ¡ d PPNT " System ÿ ´ p ¦ p – p ÿ

ÿÿÿÿ p Œ -
p " % " 9 ¡ d PPNT Century Schoolbook ,

Helvetica

. + n ¡ d PPNT Symbol , Symbol)
=) +) +) +) = ¡ d PPNT Century Schoolbook (1 (
& 1 * 2 (
9 1 * 4 (c 2 ¡ d PPNT MT Extra (L L ¡ d PPNT Century Schoolbook) . ¡ d PPNT " System ÿindeligNormalVanlig tekstøâéì•W €ý 1990-91 Apple Computer Inc. ý 1990-91 Bitstream Inc.âêÇÓêCopyright 1990-91 Apple Computer Inc. Copyright 1990-91 Bitstream Inc.NormaaliåÊæ ÓÇÏçCopyright 1990-91 Apple Computer Inc. Copyright 1990-91 Bitstream Inc.1.0 @ , ° %Id°@QX ÈY!-,° %Id°@QX ÈY!-, ° P°
y ¸ÿÿPX Y° ° % ° %#á ° P°
y ¸ÿÿPX Y° ° % á-,KPX ¸ CEDY!-,° %E`D-,KSX° %° %EDY!!-,ED- þ –
þ²þ² á ! ! ! ! f ½ z H ã Á N ó C ƒ ® Î c · F Ø y ñ \
Â
Q
Ã
ù
S , < E Y] d e g
M \ g h i j Ü â ä ê î ô á í s t “
™ Æ È Í Ï â í ò ý
Ô
Ö
Ú
ä

:

;

J

S

?

@

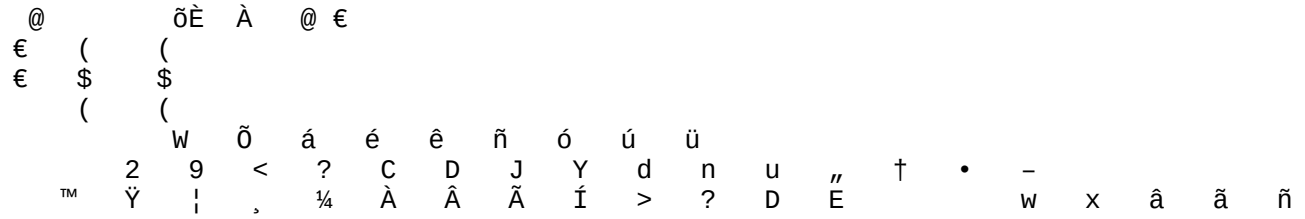
D

E

```
}
```

,

…) 1 2 3 9 ; B D K N O P R ^ _ b d h
j k s z ~ , ’ “ –
ž £ µ Æ Õ úô îèîç ææäçî æ æ æ æ æ æ æ æ æ æ æ æ Ý ç æ æ æ æ ä äç
ä ä äç ç æ ä äç äææç ä ç äç ç



9 : o p ‡ ˆ 1 2 3 4 5 ò ó ö ø i j ò ó ÷ ø

! / m n r sþ þ þ þ ý þ þýþü þ þ þ
ý þý þ þ ý ý ù ù ù ù ò ý þ ü ü ü þ þ üì þ ü ü þ ü ü ü ü ü üì üì ý þ ü

@ ýp € @ € À] s ¶ · ¸ ¹ ½ æ ç è é ê [ ]
Ð Ñ Ò Ó Ô ú ý ' (V [¢
t z ç è
ö ! !b !c !o !p !‰ ! "ï "ð "ñ "ò "ö # # # # # #> #A %
%
&ö &÷ ' ' (_ (b (¥ (¦ (½ (¾ (ã (ä (ö (÷)))9):)F)G
)H)I)_)`)a)), þ ý÷ þ ý÷ þ þ ý÷ ÷ ý ý ý ý ý ð ï í ê ê þ ý÷ þ ý÷ ÷
ý ã ï ÷ ý ý Ü ý Õ Ï Ï Ì ýÏ ý

ü

B [2] TP

B ö QÀ

@ 0¤ € @ €

@ ″

@ À P),)ƒ)œ))ž)·)¸)â)ã *5 *6 * * *œ * *ž *¶ *¼
+ + + +Ž +Ó +Ô - - - - - - -) -* -+ -, -- . .
.‡ . /] /f /n /o /u /w /} / /€ /, /‘ /’ /™ / /« /´ /À /Ê
/Ë /Í /ê /ñ /ô /õ /ö /ÿ 0 0$ 0' 0(0) 00 08 0= 0> 0G 0L 0N
0Y 0b 0j 0k 0q 0s 0y 0{ 0| 0~ 0‰ 0‹ 0Œú ùú ù ù ò ð íì í í í í ë ù
ù ë ùå å ù ë ë ë ë ì ìëìë ëì ëìëìëâëìëìëâë ë ë ì ë ë ë ë ì ë À

À € € @

B ô Vü @

ü

Y 0Œ 0Ž 0” 0é 0í 0ï 0ü 1 1x 1y 1Ï 1Ñ 2_ 2h 2½ 2¾ 2Ì 2Ý 3W 3
` 4M 4V 4s 4u 4 4 4‘ 4’ 4– 4—
4´ 4½ 4å 4ê 5È 5Ñ 6B 6I 6É 6Ê 7

7
7

7	7	7x	7	8	8	8"	8#	8<	8B	8Å	8Ç	8ô	8ý	9ª	9³	9À	9Â	9Ô	9Þ
:ý	:ÿ	;ì	;õ	<	<	<¿	<Á	=0	=5	=6	=I	=J	>T	>[	>\	>Š	?	?7	@

@

@ @Â @Ä @É @Î @×þ þ ý ý ü ü ü õ ý ü ü ü þ ü ü ü ü ü ü ü þ üï ü í ê ê ç ê
ê ç ê ç ü ç ç êä í üáü ý çü çü ü @ € @

```
 @        Xø  @ € À Y  @×  AÙ  Aâ  A÷  Aù  Aý  B°  B²  B¼  CG  Cw  Cˆ  C‰  CŠ  C
‹  CŒ  CŽ  C    C    C‘  C’  C£  C¤  C¥  C¦  Cª  C¬  C-  C®  C¯  C²  CÂ  CÃ  CÄ  C
Å  CÉ  CÊ  CÌ  CÍ  CÎ  CÏ  CÐ  Cß  Cý  DL  Dm  Ds  Dt  Dy  Dz  D   D€  D   DŽ  D
–  D—
  D   Dž  D÷  E   E>  ED  EF  EL  EQ  EW  Eø  Eù  F   F   F   F   F$  F'  F1  F4
  G     G
```

G@ GC GW GZ Gä Gí I I
ID IE IS I` Jd Jq Lö þ ûþ ûþ þ ú þô ú þô ú þô ú þô ú þ þ ú þô ó þ û û û
û û û ó þ þ þ ú ú þ ô ô ô ô ô ð ð é ó ð

@ j(€ €

À @ \ Lö M M M! M1 M9 M: MB MO MQ Mu M¬ Mµ MÂ MÓ Nž N
N¹ NÆ O O O> OB O` Od OÒ Oß P) P, Pk Px P³ Pµ PÙ PÚ PÜ Q
Q Q3 Q6 Q Q QÇ QÈ Sâ Sï Sú Sü T% T' U U- UP U¨ U± VZ V\
V’ Xg Xl Y Y Y Y_ YE YF YN \< \> ^ ^
^Y ^\ ^š ^œ _/ _D _¯ _¼ _Ö _× _ç `B `C `O `P `iþ þ þ þ ûþ ø ø û
ø ø ø ø ø ø ø ûþòþ ø ø ì å ø ì ø ûþ ø ûþ ø ì ûþâþ þ þ þ þ þ ßÞÛÞ Ù ø @ €
€ @

B î ,˜

@

€ @ V `i `p `¬ `¹ `Õ `Þ a a a-
a) a] ag a’ aš cl cm cq cr cÑ cÒ cÓ cÔ câ cå dÂ dÅ e- e° eÕ
eÙ f

f fh fn fÙ fÚ fé fþ g g€ gÌ gÍ iÕ iÖ iä j k- k-
nì ní nú o o o- o, oO o± o² o¾ o¿ oÙ oß oä oê p+ p/ p< p@
pB pG pL pR pš pž p¼ pÀ pÓ p× pÚ pä q q! qK qSý ý ý ý ý ý ý ü
û ü û õ ý ý û û û î í ê ê ã í ê Ü í Õ í Ó ý ý ý ý ý ý ý ý ý ý ý ý ý @

@ ¦€

@ Š¼

@ l €

@ ”

@ À € S qS r r, r† r‹ s` sa se sf t! t$ t¨ t« tç tê uã uï
| |
|± |³ |È ~¶ ~¸ ~Ï M N \ ~ ƒ „ ’ µ , , ,# ,% ,> ,D
,I ,O , ,” ,¡ ,¥ ,§ ,¬ ,± ,· ƒ ƒ
ƒ ƒ ƒ! ƒ& ƒC ƒG ƒo ƒw ƒ© ƒ- ƒÂ ƒÊ ƒÒ ƒÜ „& „1 „þ „z „Á „Ì
„Ù „á …H …L …ÿ † † † †# †$ †% †& †r †u ŠW Š[Šº Š½ þ þ ý þ
÷ ÷ ô þ ô ñþ ñþ ê é â é à ô þ ý þ ý þ ÷ þ þ
@

@ Å´ €

@ °Ü €

À @ X Š½ ‹u ‹y ŒÒ ŒÕ ŒØ ŒÙ Œç Œþ W X d f … Š Ù
á Ž* Ž2 Žµ Ž¿ Žò Žü C I é ñ } ƒ Œ –
Ú â ‘î ‘ð ‘ý “¤ “¦ ”q ”s ”x • • •z •| –Å –Ç —¢ —¤ —¥ —
¦ —¨ —¬ ˜Â ˜Ä ˜Î š

š> šL šM šQ šR šS šT šj šk šo šp šr šs št šu š‹ šŒ š š‘ š“
š” š˜ š™ š© šª š® š¯ š± š² š³ þ ø ñ ð î ë ë ë ë ë ë ë ë ë ë ë ë èþ è
èþ è è è èá èþ èþ þ à à þ à þ à þ à þ à þ à þ à À

B ê :\ € @ €

@ ßT

@ X š³ š´ šÄ šß ›1 ›3 œr œ œö # Ÿ% Ÿa Ÿn Ÿu Ÿ, Ÿˆ Ÿ ¹ ŸÁ ŸÏ ŸÕ

 % ó> óA óU ót ó§ óª óí ô   ô* ô-
ôj ôˆ ô± ô´ ôø õ   õ. õ1 õ| õ–
õ³ õ´ õ· õ¸ õ¼ õ½ < þ ý ÷ þ ýôý þ þ þ þ þ ý ÷ þ ÷ þ ÷ þ ÷ þ ÷ þ ò ò ì

@   @ €

€ @ 2 -) * + , < E F G [
k ƒ — ª É Þ ó) = \ r ¨ À Ô í

+ C X Y Z \ ] e f g »

\ h i k { z

:ûûûûûûûûöööðëöööæææáááæáááááæáááááááæÜÜÜ×ÒöööÍÈöÀ¸ öö°«¦¡œ– $

$

$ $

ÿ $ ð x $

$ @ $
$ @ $
$
$ $ è $
é $
é $ $
ô $:

:

<

=

S

T
f

R Y n ‡ • ¢ ¨ Ç È
2 J Y u „ ‹ ™ ® ¼ Â Ã Þ ß ì „ … ˜ ¥
¦ùòìçâÝçÉµµµµµÉìç§“”””””””””” çyçtoçiic Ð $
Ð $
$ $
$
- @ @ @! @$ ´
P
Ð $
- @ @ @! @$ ´
P
Ð
$
- @ @ @! @$ ´ $
- @ @ @! @$ »
P
Ð $
- @ @ @! @$ »
P
Ð $
$
$
$
$
$ ‰ & ¦ â ä å
-
" $ & ( ) , . 1 4 8 9 = ? E K R S Y \ c
j u v } € Š ” ¢ £ - ° » Æ Ø Ù ä ç ó
ûôôîéûûßßßßß´ßßßßß‰ßßßßß´ßßßßß´ßßßßß´ßßßßß´ßßßßß´ßßßßß* ’ ” l˜ @ ÿ” '
ã
õ ¹-. @ @ @ @ @ @ @ @ @ @ @ @ * ’ ” l˜ @ ÿ
” ' ã
õ ¹-. @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @
ÿü$ » 6
/ 0 Ñ Ú 7 * 5 G ÿ - é £ ç
é ! ! " " "& #C #J $œ &ö &ø ' ' (â (ã (å (æ))))
*5 *7 *F ,; ,L ,M .
. /O /P /, /‰ /« /Ë /ý 00 0H 0N 0O 0~ 0… 0” 0± 0ÍÕÏÊÊÅÊÊÀ»ÊÅÊ
ÊÊÊ´ÏÅÊÊ»ÊÄÊÊ´ÏÊÊÊ®ÊÊÊ®ÊÊ®ÅÊ»ÊÊÅÊÊšššššššššššššš - @ @ @! @$ ´
P
Ð þ ÿ
* ’ ” l˜ @ ÿ” ' ã
õ ¹-. @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ < 0Í 0é 0ï 1 1 2
½ 2¿ 2Þ 2ß 4œ 7± 7À 8Å 9À :ý < <¿ = >S >T >U >_ >i >j >p >
w >} >„ >Š >‹ >Ž >’ >• >™ >œìæááÚæÔááÏáÄÄÄÄÄá¼³©©Œ,,,,,Y,,,,, (
” l˜ @ ÿ” ”
” @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @
ÿü$ » ” l˜ (ÿ”
@ @ @ @ @ @ @ @ @ @
ÿü$ » ÿü$ » x x
Ð þ˜ x x
- @ @ @! @$ ´
P
Ð " >œ > >¡ >§ >« >¯ >³ >´ >º >Á >È >Î >Ò >Ó >Ú >â >ê >ð >ô
>õ ?7 ?8 ?U @

```
  @Â  A÷  B°  CG  CN  CY  Cf  Cv  Cw  C†  Cˆ  CŽ  C”×ÍÍÍÍÍ¤ÍÍÍÍÍ¤ÍÍÍÍÍ¤ž™”™‰‰‰‰€
€€€]€€€€  "          ” l˜ 4 ÿ” ?
Ò 1 ›   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @           ÿü$ »
         Ð þ˜  x  x   þ                              (          ” l˜ @ ÿ”   ”
  ”     @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @
                  ÿü$ »  (           ” l˜ @ ÿ”   ”
  ”       @ @ @     @ @ @     @ @ @     @ @ @     @ @ @$  C”  C•  C   C£  C¬  C´
  Cµ  C¿  CÂ  CÌ  CÒ  CÓ  Cý  Cþ  Cÿ  DK  DL  DR  D\  Db  Dl  Dm  Dp  Dv  D|  D,
  Dƒ  D‡  D   D™ÝÔÔÔÔ±ÔÔÔÔ±«¦¦¦¦œœœœyooooLooo   "          ” l˜ 4 ÿ” u T
```

A ; @ @ @ @ @ @ @ @ @ @ @ @
ÿü$ » " ” l˜ 4 ÿ” u T

```
A ;   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @
                  ÿü$ »                          "           ” l˜ 4 ÿ” ?
Ò 1 ›   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @           ÿü$ »  "           ” l˜
 4 ÿ” ?
Ò 1 ›     @ @ @     @ @ @     @ @ @     @ @ @   D™  D   D¡  D§  D±  Dº  DÃ  DÄ
DË  D×  Dá  Dê  Dë  E   E   E#  Gy  G€  ID  IF  I`  Ia  K(  MO  NB  N]  Nž  P³
R!  R3  RX  R|  R}  R   Rº  Rà  Rá  Rô  S   SH  SI  S}  S™  SÒ  Ta  T‹  TŒ  Tª
TÔ  Tð  U   U   UÍ  Uò  Uó  V   V,  VJ  VX  VZ  W¶  WÛöÓöööÓööööÓÍÈÃ¾¹¾²Í¾¾-§§
-œ§§§§§§§§§§§§§§œ§§§§§§§œ§§§§§§§œ§
         Ð þ˜  x  x
                                                  þ         ÿ
"           ” l˜ 4 ÿ” u T
```

A ; @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @
ÿü$ » = WÛ Wõ Wö X X/ XW Xe Xg Y ZT Z~ Z˜ Z™ Z
¹ ZÎ Zè Zé [[$ [/ [B [a [€ [[• [–
[· [ë \ \/ \1 \2 _. _/ _4 _9 _> _C _D _F _J _N _P _Q _S _W
_[_] _^ _` _d _h _j _k _m _q _v _x _y _| _ úúúúúúúïïúúúúúúúúúúú
úúúúúúúúúúêêêàààà½ààààšàààà½àààà½àààà½àà " ” l˜ 4 ÿ” j ®
Ã @ @ @ @ @ @ @ @ @ @ @ @ " ” l˜ 4 ÿ” j ®
Ã @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ ÿü$ »
Ð þ˜ x x < _ _„ _† _‡ _Š _ _’ _” _• _˜ _› _
_¢ _£ _è _÷ aÆ aÝ d d

fØ fÙ fÛ fÜ fþ fÿ g h iÒ iÓ iÔ iÕ i× j j kf l nì nî o
o o-
oO o^ q~ q‘ si tR tY t- tÔ u u> u y yŒ { |± ~¶ €] H I
J K L MööÓöööööÓööööÓÍÈÃ¾ÃÈÃÃ··Í·ÃÃÃÃÃ°ÍÈÃÃÃ°ÍÃ·ÍÈÃ¾ÃÃÈÃ¥¥¥¥ÃÈÃÃ¥¥Ã····
.
Ð þ˜ x x þ
" ” l˜ 4 ÿ” j ®
Ã @ @ @ @ @ @ @ @ @ @ @ @ @ @ @
ÿü$ » A M O ~ € , ƒ … ¶ · Ç ƒÑ …

… †w †~ ˆ€ ‰’ ‹{ Œ× ŒØ ŒÚ Œþ
 f Ž ’
‘& ‘ ‘î “ “(“B “Y “Z “y “z “’ “” “¤ ” ”$ ”7 ”8 ”W ”X ”o
”q • •z –Å —¢ —¦ ˜Â š

š š š-
š=ùóùùùùùùóíèããÞãèãããããùóèãããÞããÓËËËËËËËÓËËËËËËËÓÀÀÀ´ÓÓ«««««
ÿü$ »
Þ˜ x x
Þ˜ x x Þ˜
Ð Þ˜ x x Þ
; š= š> šI šL šQ šV šW šg šj šr šw šx š‡ š‹ š“ š› šœ š
¦ š© š± š¶ š· šß šà œq œr œy œ€ œ‡ œŽ œ œš œ œŸ œ¢ÝÔÔÔÔ±ÔÔÔÔÝ
ÔÔÔÔÝÔÔÔÔÝ«¦¦¦Ôœœœ yÔœœœ
" ” l˜ 4 ÿ” W
ù

Ú | @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @
ÿü$ » " ” l˜ 4 ÿ” W

ê I ³ @ @ @ @ @ @ @ @ @ @ @ @ ÿü$ » " ” l˜
4 ÿ” W

ê I ³ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @" œ¢ œ£ œ³ œ¶ œ¹ œ¼ œ½ œÌ
œÏ œÒ œÕ œÖ œà œã œæ œé œê # $ Ÿ$ Ÿ% Ÿ& Ÿ, Ÿ7 ŸG ŸV Ÿ` Ÿa
Ÿb Ÿe Ÿg Ÿi Ÿk ŸmÝÔÊÊÊ§ÔÊÊÊ§ÔÊÊÊ§¡œœœÔÔÔÔÔmÔÊÊÊÊÊ
. ” l˜ L ÿ” ã
O ù ±
+ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @
" ” l˜ 4 ÿ” W
ù

Ú | @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @
ÿü$ » ÿü$ » " ” l˜ 4 ÿ” W
ù

```
Ú |     @ @ @     @ @ @     @ @ @     @ @ @!  Ÿm  Ÿn  Ÿu  Ÿy  Ÿ{  Ÿ}  Ÿ   Ÿ   Ÿ,
  Ÿˆ  ŸŽ  Ÿ   Ÿ’  Ÿ”  Ÿ—
  Ÿ˜  Ÿ™  Ÿ   Ÿ¢  Ÿ¥  Ÿ¨  Ÿ«  Ÿ¬  Ÿ-  Ÿ°  Ÿ²  Ÿ´  Ÿ¶  Ÿ¸  Ÿ¹  ŸÁ  ŸÅ  ŸÇ  ŸÊ  ŸÌ
  ŸÎ  ŸÏ  ŸÕÑÈ¾¾¾¾¾ È¾¾¾¾¾¾ È¾¾¾¾¾¾ È¾¾¾¾¾¾`È¾¾¾¾¾¾ È   .           ” l˜ L ÿ”   ã
O ù ±
+   @ @   @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @ .           ” l˜ L
 ÿ”   ã
O ù ± +     @   @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @
                ÿü$ »             ÿü$ »  .           ” l˜ L ÿ”   ã
O ù ±
+     @   @     @ @ @     @ @ @     @ @ @     @ @ @     @ @ @%  ŸÕ  ŸÛ  ŸÝ  Ÿã
Ÿå  Ÿè  Ÿé  Ÿê  Ÿñ  Ÿó  Ÿú  Ÿü  Ÿÿ       %   6   M   N   ]   s   t       ‘   ±
 É   Ê   ç   ¡   ¡"  ¡5  ¡Z  ¡j  ¡k  ¡™  ¡Ç  ¡þ  ¢   ¢+  ¢1  ¢4  ¢G  ¢H  ¢W  ¢m
¢n  ¢Š  ¢‹öööööÇ¾ööööö ‰„ zzzzzzzzzzzzzzzzzzzzzzz uzzzzz                    þ
      ÿ                    .           ” l˜ L ÿ”   ã
O ù ±
+     @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @           ÿü$ »
.           ” l˜ L ÿ”   ã
O ù ± +     @   @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @
                ÿü$ » .  ¢‹  ¢ª  ¢½  ¢Ô  ¢Õ  ¢ô  £   £3  £4  £X  £m  £x  £ˆ  £
—  £È  £Ú  £å  ¤   ¤   ¤(  ¤.  ¤/  ¤C  ¤D  ¤w  ¤ª  ¤Ý  ¤ó  ¥   ¥-
  ¥`  ¥}  ¥®  ¥¼  ¥Ô  ¥Þ  ¥ß  ¦   ¦   ¦   ¦   ¦/  ¦0  ¦?  ¦U  ¦V  ¦r  ¦s  ¦   ¦¬
  ¦Å  ¦Æ  ¦è  §
  §-  §.  §c  §~  §“  §£  §±  §²  §Â  §Î  §Þ  ¨
```

¨? ¨Z ¨j ¨{ ¨‹ ¨ ¨£ ¨¤ ¨Ä ¨Ò ¨â ¨ã ¨ñ ¨ó ¨ô © © ©- ©= ©\
©| ©} ©“ ©” ©» ©Ø ©ø ª

ª ª ûûûöñûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûû
ûûûûûûûûûûûûûûûûûûûûûûû þ _ ª ªA ªa ªb ª ª¼
ªÜ ªþ « «' «I «a «k «q «t «, «ƒ «— «Ä «Å «û ¬ ¬
¬" ¬: ¬S ¬T ¬ ¬ª ¬Ñ ¬î ¬ô -(-G -p -Ž -″ -- -—
-É -þ ®/ ®; ®T ®U ®o ®‰ ®¤ ®¥ ®Ä ®ô ¯ ¯1 ¯J ¯K ¯e ¯ ¯€ ¯š
¯› ¯Ø ¯ý °* °N °O °} °˜ °¬ °Ä ± ±A ±d ±‹ ±¦ ±Á ±Ï ±Ð ±ø ²
²/ ²0 ²v ²š ²¿ ²ë ³ ³% ³D ³Y ³~ ³¥ ³Ó ³å ´ ´ ´ ´" ´$ûûûû
ûûûûûûûûûûöûû
ûûûûûûûûûûûûûû þ a ´$ ´% ´8 ´9 ´M ´` ´a ´' ´ ´½ ´¾
´Õ µ µ? µG µH µd µe µw µŠ µ‹ µ© µÃ µæ µè µé ¶ ¶ ¶1 ¶2 ¶]
¶v ¶¡ ¶Û ¶ì ¶ò ·

·) ·; ·O ·] ·_ ·` ·z ·Œ ·§ ·¨ ·¼ ·È ·á ·ü ¸& ¸6 ¸K ¸Q ¸e ¸f
¸i ¸- ¸æ ¹ ¹
¹-
¹R ¹S ¹a ¹c ¹d ¹~ ¹‹ ¹ª ¹Ó ¹è ¹ö ¹ø º º º' º; º< ºl º{ º—
º´ ºµ ºÌ ºý »/ »\ »–
»ž »Ÿ »Ú »Û »ö ¼ ûöññ
ññññññññññññññññññññöûññññññññññññññññññññññ þ _ ¼ ¼
¼K ¼ ¼´ ¼µ ¼Î ¼ø ½8 ½I ½O ½c ½w ½Œ ½ ½§ ½» ½Ë ½á ¾-
¾9 ¾m ¾™ ¾Ÿ ¾· ¾¸ ¾Ó ¾â ¿

¿* ¿7 ¿V ¿_ ¿q ¿r ¿ ¿ƒ ¿ž ¿¯ ¿° ¿Ò ¿ô À À ÀJ Àk Àl ÀŠ À·
ÀÆ ÀÑ Àç Á Á Á Á9 ÁR Ág Áp Á† Á‡ Á° ÁÒ Áå Â Â(Â5 ÂW Â`
Âr Âs Âš ÂÁ Âú Ã

Ã' ÃQ Ãn Ãš Ã¹ ÃÐ Ãö Ä Ä, ÄR Ä_ Äu Ä‡ Ä” Ä¦ Ä¬ Ä® Ä¯ ÄÈ ÄÉ
Äí Å Å6 Å7 ÅQûû
ûû c ÅQ Ål Å” Å² Å¿ Åõ Æ%
Æ4 Æ6 Æ7 ÆN ÆO Æc Æv Æw Æ§ Æ¶ ÆÒ Æï Æð Ç Ç8 Ç9 ÇP Ç Ç³ Çà
È- ÈX È` Èa ÈŸ ÈÉ ÈÊ É É É
É$ ÉD ÉE Éy É- Éâ Éã Éü Ê& Ê] Ên Êt Êˆ Ê› Ê¯ ÊÂ ÊÖ Ê×
Êù Ë
Ë Ë: Ëx Ë’ ËÆ Ëò Ëø Ëù Ì Ì(ÌR Ìp Ì} Ìœ Ì¸ ÌÃ ÌÕ ÌÖ Ìê Ìù
Ìû Ìü Í Í ÍC Ím Í–
ÍÂ ÍÃ Íè Î Î= Îq Î¡ ÎÄ ÎÅ Îè Ï Ï4ûûûûûûûûöñûûûûûûûûûûûûûûûûûûûû
ûû þ
_ Ï4 Ïd Ï‹ Ï Ï¡ ÏÆ Ïó Ð! ÐD Ðg Ðy Ðz Ð£ ÐÍ Ðý Ñ-
Ñ; ÑL ÑM Ñz Ñ© ÑÌ ÑÍ Ñð Ò Ò< Òl Ò“ Ò¨ Ò© ÒÎ Òú Ó ÓA ÓS Ó}
Ó- ÓÜ Óê Óô Óú Ô Ô Ô Ô1 Ô2 ÔQ Ôp Ô Ô‘ Ô© Ôª ÔÌ Ôä Õ Õ
Õ5 Õ^ Õs Õ• Õ¶ ÕÃ Õå Õò Ö Ö
Ö
Ö" Ö3 ÖT ÖV ÖW Ös Öt Ö• Ö¶ ÖØ ÖÙ Öð Öñ × ×* ×Y ×Z ×} ×¦ ×»
×Þ Ø Ø
Ø. Ø; ØM ØS ØT Øk Ø} Øž Ø Ø¡ûûû
ûû c Ø¡ Ø¼
ØÍ ØÎ Øð Ù Ù5 Ù6 ÙV ÙW Ù„ Ù® Ù½ ÙÊ Ú Ú ÚI ÚO ÚP Ún Ú† Ú›
Ú¤ Úº Ú» Úä Úÿ Û Û4 ÛU Ûb Û„ Û ÛŸ Û ÛÃ ÛÞ Û÷ Ü Ü Ü Ü$
Ü% ÜO Üy Ü£ ÜÎ ÜÏ Üü Ý ÝH Ýk Ý Ýµ ÝÝ Þ Þ ÞX Þx Þ—
Þ¬ ÞÜ ß ß, ßQ ß{ ß ßº ßä à
à5 àO à] àn à• à¸ àÚ á á+ áP á^ á¥ áÅ áä áù â(âU ây â âÆ
âØ ã ã/ ãW ã€ ãš ã¨ ã² ã¸ ãÎûûû
ûû c ãÎ ãÐ
ãÑ ãê ãë ä ä3 äX äY äs ä äµ äÓ äà å åF åU åW åX åj åk å~
å’ å“ åÅ åÔ åð æ
æ

æ æ6 æ: æN æO æf æ˜ æØ æà æá æò ç$ ç` çl çm ç© çª ç¿ çÀ çÒ
çÝ çÞ çý è è< è= è‰ è¾ èÏ èÕ èù é é5 é7 é8 éS éh é† é“ é”
éÈ éü ê1 ê2 êT êf ê’ ê· êç ë ë ë# ë: ëp ëq ë‹ ë ëÉ ëû ëü
ì! ìZ ìw ìš ì ì¡ ì¹ûûûûûûûûûûûûûûûûûöñûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûû
ûûû þ _ ì¹
ìÝ í í< íM íS íg íh í‡ í- íÜ íÿ î î î î î1 î< î^ î_ î
î¿ îð îñ ï ï% ïQ ïv ï¦ ïÅ ïÜ ïâ ïù ð-
ð. ðP ð| ð« ðÚ ðà ðá ðï ðð ñ

ñ. ña ñ{ ñ} ñ~ ñ—
ñ¢ ñº ñ» ñß ò ò(ò) òU òz òª òÉ òÏ òæ ó ó, ó. ó> ó ó¦ ó§
ô ô) ô* ô° ô± õ

õ-
õ. õŒ õ² õ³ õµ õ¶ õ· õ¹ õº õ» õ¼ õ½ûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûû
ûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûûöñññññññññññññññûêåñêåñññû ó $
ó ÿü ÿ X —
ÿÿÿÿÿÿÿÿÿ ÿÿ Body Text caption
SubHeading Signature

endnote text
PseudoCode
InlineCode
List Bullet 4
List Bullet
List Number List ÷

€ ÿÿÿÿÿÿÿÿÿÿ
€ 4 4ÿÿ
@
@
@ À €
€
€ ((
€
€
€
€ á € $ D â $ D ã $ D ä
° $ D å À $ D æ Ð $ D ç à $ D è
õ x $ D é ð x $ Dÿÿÿÿÿÿÿÿÿÿ ô à!À @ÿÿ
÷ ð <
ø ð <
ù ð <
ú ð <
û ð < ü ð < ý ð < þ ð < @ ÿ
ð h
x
x x
ð x
à Ð

P
Ð
þ˜ Ð þ˜ x x
h þ˜
h þ˜ + ó ô ö Þ
< Z k ô½ õ½ õÂ ÿÿÿÿ õÅ ; < Z [j k
ô½ ÿÿ ÿÿ ÿÿ ÿÿ ÿÿ ÿÿ ÿÿ @ ÿÿ
Õ s), 0Œ @× Lö `i qS Š½ š³ < ! " # $ % & ' ( )

: ¦ 0Í >œ C” D™ WÛ _ M š= œ¢ Ÿm ŸÕ ¢‹ ª ´$ ¼ ÅQ Ï4 Ø
¡ ãÎ ì¹ õ½ * + , -
. / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A Ò&p?ÿ H H Ø (ÿá
ÿâ ù F G (ü H H Ø (d ' h
=à/Ð hV @ € H -:LaserWriter 8
Times Helvetica Courier Symbol

```
}
```

B_Toc347461014

B_Toc347460760

B_Toc347460695

B_Toc347460659

B_Toc347460631

B_Toc347460558

B_Toc347471499

B_Toc347471389

B_Toc347471856

B_Toc356911628

B_Toc356914183

B_Ref357084512

B_Toc357133632

B_Toc357145830

B_Toc357147857

B_Toc357148473

B_Toc357150105

B_Toc357764613

B_Ref357150559

B_Ref357074685

B_Ref357075750

B_Ref357075769

B_Ref357076201

B_Ref357076383

B_Toc356911629

B_Toc356914184

B_Toc357133633

B_Toc357145831

B_Toc357147858

B_Toc357148474

B_Toc357764614

B_Toc357150106

B_Toc356911630

B_Toc356914185

B_Toc357133634

B_Toc357145832

B_Toc357147859

B_Toc357148475

B_Toc357764615

B_Toc357150107

B_Ref357084306

B_Toc356911631

B_Toc356914186

B_Toc357133635

B_Toc357145833

B_Toc357147860

B_Toc357148476

B_Toc357764616

B_Toc357150108

B_Ref357084721

B_Toc356914187

B_Toc356911632

B_Toc357133636

B_Toc357145834

B_Toc357147861

B_Toc357148477

B_Toc357764617

B_Toc357150109

B_Ref357130977

B_Ref357131004

B_Ref357133053

B_Toc356911633

B_Toc356914188

B_Toc357133637

B_Toc357145835

B_Toc357148478

B_Toc357147862

B_Toc357764618

B_Toc357150110

B_Ref357133341

B_Toc356911634

B_Toc356914189

B_Toc357133638

B_Toc357145836

B_Toc357147863

B_Toc357148479

B_Toc357764619

B_Toc357150111

B_Toc356911635

B_Toc356914190

B_Toc357133639

B_Toc357145837

B_Toc357147864

B_Toc357148480

B_Toc357764620

B_Toc357150112

B_Ref357132089

B_Ref357133421

B_Toc356911636

B_Toc356914191

B_Toc357133640

B_Toc357145838

B_Toc357147865

B_Toc357148481

B_Toc357764621

B_Toc357150113

B_Ref357132123

B_Ref357132186

B_Ref357132229

B_Ref357132268

B_Toc356911637

B_Toc356914192

B_Toc357133641

B_Toc357145839

B_Toc357147866

B_Toc357148482

B_Toc357764622

B_Toc357150114

B_Ref357132329

B_Ref357132353

B_Toc356911638

B_Toc356914193

B_Toc357133642

B_Toc357145840

B_Toc357147867

B_Toc357148483

B_Toc357764623

B_Toc357150120

B_Ref357132373

B_Toc356911639

B_Toc356914194

B_Toc357133643

B_Toc357145841

B_Toc357147868

B_Toc357148484

B_Toc357764624

B_Toc357150121

B_Ref357133499

B_Ref357132848

B_Ref357132815

B_Toc356914196

B_Toc357133645

B_Toc357145842

B_Toc357147869

B_Toc357148485

B_Toc357764625

B_Toc357150122

B_Toc356914197

B_Ref357084543

B_Ref357084607

B_Toc357133646

B_Toc357145843

B_Toc357147870

B_Toc357148486

B_Toc357764626

B_Toc357150123

B_Toc356914198

B_Toc357133647

B_Toc357145844

B_Ref357146615

B_Ref357146659

B_Toc357147871

B_Toc357148487

B_Toc357764627

B_Toc357150124

B_Toc356914199

B_Toc357133648

B_Toc357145845

B_Ref357146744

B_Ref357146784

B_Toc357147872

B_Toc357148488

B_Toc357764628

B_Toc357150125

B_Toc356914200

B_Toc357133649

B_Toc357145846

B_Ref357146897

B_Ref357146953

B_Toc357147873

B_Toc357148489

B_Toc357764629

B_Toc357150126

B_Toc356914201

B_Toc357133650

B_Toc357145847

B_Ref357147052

B_Ref357147108

B_Toc357147874

B_Toc357148490

B_Toc357764630

B_Toc357150127

B_Ref357147631

B_Toc357147875

B_Toc357148491

B_Toc357764631

B_Toc357150128

B_Toc356914202

B_Toc357133651

B_Toc357145848

B_Ref357147724

B_Ref357147741

B_Toc357147876

B_Toc357148492

B_Toc357764632

B_Toc357150129

B_Toc356914203

B_Toc357133652

B_Toc357145849

B_Ref357147788

B_Ref357147806

B_Toc357147877

B_Toc357148493

B_Toc357764633

B_Toc357150130

B_Toc356911640

B_Toc356914195

B_Toc357133644

B_Toc357145850

B_Toc357147878

B_Toc357148494

B_Toc357764634

B_Toc357150131 < < G n n n n n n n n n n =

¨ Ã å - - - - - - - - 8 8 8 8 8 8 8 8 -
é ! ! ! ! ! ! ! ! %ø +> +> +B +B +B +B +B +B /ï 1¿ =
õ >; >; >; >; >; >; >? >? BÓ D D D D D D D D D D D
D D D D D HF ^£ `É `É `É `É `É `É `É `É eÜ h× mî n-
p p p p p p p p €O €… ″ ″ ″ ″ ″ ″ ″ ″ ‹Ú
™ ˛ ›ê Ÿ Ÿ(Ÿ(Ÿ(Ÿ(Ÿ(Ÿ(Ÿ(Ÿ(Ÿ9 Ÿ9 Ÿ9
Ÿ9 Ÿ9 Ÿ9 Ÿ9 Ÿ9 Ÿ9 ¡7 ¡7 ¡7 ¡7 ¡7 ¡7 ¡7 ¡7 ¥ ¥ ¥ ¥ ¥
¥ ¥ ¥ ¥
ª w ª w ª w ª w ª w ª w ª w ª w ª w ³ (³ (³ (³ (³ (³ (³ (³ (³ (˛ û ˛ û
˛ û ˛ û ˛ û Å: Å: Å: Å: Å: Å: Å: Å: Å: ä[ä[ä[ä[ä[ä[ä[ä[
ä[ò1 ò1 ò1 ò1 ò1 ò1 ò1 ò1 ô½ ÿÿ d ÿÿ P
ÿÿ < ÿÿ (ÿÿ ÿÿ ÿÿ Ð
ÿÿ ¼ ÿÿ ð ÿÿ
<
ÿÿ
P
ÿÿ
d

ÿÿ
x
ÿÿ
Œ

ÿÿ
ÿÿ
´ ÿÿ
È ÿÿ
ð ÿÿ
Ü ÿÿ ì ÿÿ $ ÿÿ %` ÿÿ ;

```
   ÿÿ      F     ÿÿ      J„    ÿÿ      J˜    ÿÿ      J¬    ÿÿ      JÀ
 ÿÿ      JÔ    ÿÿ      Jè    ÿÿ      K    -ÿÿ      Jü    ÿÿ      K
!ÿÿ      K´   "ÿÿ      KÈ   #ÿÿ       KÜ   $ÿÿ       Kð   %ÿÿ       L
'ÿÿ      L,   &ÿÿ      L    (ÿÿ       sà   )ÿÿ       x    *ÿÿ       x
+ÿÿ      x(   ,ÿÿ      x<   -ÿÿ       xP   .ÿÿ       xd   0ÿÿ       xŒ
/ÿÿ      xx   1ÿÿ      ŸP   3ÿÿ       ¬t   2ÿÿ       ¬`   4ÿÿ       ¬Œ
5ÿÿ      ¬    6ÿÿ      ¬´   7ÿÿ       ¬È   9ÿÿ       ¬ð   8ÿÿ       ¬Ü
:ÿÿ      ´œ   ;ÿÿ      ÈH   <ÿÿ       þ    =ÿÿ       þ    >ÿÿ       þ¤
?ÿÿ      Þ¸   @ÿÿ      ÞÌ   Bÿÿ       Þô   Aÿÿ       Þà   Dÿÿ       ß
Cÿÿ      ß
```

Eÿÿ	è	Fÿÿ	ìH	Gÿÿ	ì\	Hÿÿ	ìp	Iÿÿ	ì„		
Jÿÿ	ì˜	Kÿÿ	ì¬	Mÿÿ	ìÔ	Lÿÿ	ìÀ	Nÿÿ	íh		
Oÿÿ	í|	Pÿÿ	í	Qÿÿ	í¤	Rÿÿ	í¸	Sÿÿ	íÌ		
Uÿÿ	íô	Tÿÿ	íà	Vÿÿ							

Wÿÿ	.	Xÿÿ	0¸	Yÿÿ	0Ì	Zÿÿ	0à	[ÿÿ	0ô
\ÿÿ	1	]ÿÿ	1	_ÿÿ	1D	^ÿÿ	10	`ÿÿ	B
aÿÿ	Qx	bÿÿ	f	cÿÿ	tl	dÿÿ	w@	eÿÿ	wT
fÿÿ	wh	gÿÿ	w|	hÿÿ	w	iÿÿ	w¤	kÿÿ	wÌ
jÿÿ	w¸	lÿÿ	<	mÿÿ	·X	nÿÿ	½\	oÿÿ	½p
pÿÿ	½„	qÿÿ	½˜	rÿÿ	½¬	sÿÿ	½À	uÿÿ	½è
tÿÿ	½Ô	vÿÿ	!d	wÿÿ	(	xÿÿ	($	yÿÿ	(8
zÿÿ	(L	{ÿÿ	(`	|ÿÿ	(t	~ÿÿ	(œ	}ÿÿ	(ˆ
ÿÿ	6Ì	€ÿÿ	<	ÿÿ	F	,ÿÿ	Fl	ƒÿÿ	F€
„ÿÿ	F”	…ÿÿ	F¨	†ÿÿ	F¼	ˆÿÿ	Fä	‡ÿÿ	FÐ
‰ÿÿ	Gl	Šÿÿ	G€	‹ÿÿ	G”	Œÿÿ	G¨	ÿÿ	G¼
Žÿÿ	GÐ	ÿÿ	Gä	‘ÿÿ	H				

ÿÿ		Gø	′ÿÿ		Jœ	“ÿÿ		J°	”ÿÿ		JÄ	•ÿÿ		JØ	–
ÿÿ		Jì	—ÿÿ		K	˜ÿÿ		K	šÿÿ		K<	™ÿÿ		K(	
	›ÿÿ		Q¸	œÿÿ		QÌ	ÿÿ		Qà	žÿÿ		Qô	Ÿÿÿ		R
	ÿÿ		R	¡ÿÿ		R0	£ÿÿ		RX	¢ÿÿ		RD	¤ÿÿ		Z`
	¥ÿÿ		Zt	¦ÿÿ		Zˆ	§ÿÿ		Zœ	¨ÿÿ		Z°	©ÿÿ		ZÄ
	ªÿÿ		ZØ	¬ÿÿ		[	«ÿÿ		Zì	-ÿÿ		h	®ÿÿ		h
	¯ÿÿ		h(	°ÿÿ		h<	±ÿÿ		hP	²ÿÿ		hd	³ÿÿ		hx
	µÿÿ		h	´ÿÿ		hŒ	¶ÿÿ		o	·ÿÿ		o	¸ÿÿ		o0
	ºÿÿ		oX	¹ÿÿ		oD	»ÿÿ		€	¼ÿÿ		€	½ÿÿ		€(
	¾ÿÿ		€<	¿ÿÿ		€P	Àÿÿ		€d	Áÿÿ		€x	Ãÿÿ		€
	Âÿÿ		€Œ	Äÿÿ		¦Œ	Åÿÿ		¦	Æÿÿ		¦´	Çÿÿ		¦È
	Èÿÿ		¦Ü	Éÿÿ		¦ð	Êÿÿ		§	Ìÿÿ		§,	Ëÿÿ		§
	Íÿÿ		·”	Îÿÿ		·¨	Ïÿÿ		·¼	Ðÿÿ		·Ð	Ñÿÿ		·ä
	Òÿÿ		·ø	Ôÿÿ		¸	Óÿÿ		¸						

z z z z z z z z z z z z z z z z z z z z
G

² Í ï 4 4 4 4 4 4 4 4 F F F F F F F F -
ó !% !% !% !% !% !% !% !% & +K +K +K +K +K +K +K +K /ù 1É =
þ >T >T >T >T >T >T >T >T BÜ D D D D D D D D D" D" D
" D" D" D" D" D" HP ^¬ `Ü `Ü `Ü `Ü `Ü `Ü `Ü `Ü eæ há mø n) p
p p p p p p p €Y € „ „ „ „ „ „ „ „ ‹ä %
% % % % % % % ™Á ›ó Ÿ
Ÿ5 Ÿ5 Ÿ5 Ÿ5 Ÿ5 Ÿ5 Ÿ5 ŸL ŸL ŸL ŸL ŸL ŸL ŸL ŸL ŸL ¡F ¡F
¡F ¡F ¡F ¡F ¡F ¡F ¡F ¥. ¥. ¥. ¥. ¥. ¥. ¥. ¥. ¥. ª ª ª ª
ª ª ª ª ª ³7 ³7 ³7 ³7 ³7 ³7 ³7 ³7 ³7 ¹ ¹ ¹ ¹ ¹ ÅM
ÅM ÅM ÅM ÅM ÅM ÅM ÅM ÅM äi äi äi äi äi äi äi äi äi ò= ò= ò=
ò= ò= ò= ò= ò= ò= ô½

Á € € E õ

E ˜ NETWORK
ADMINISTRATOR