CSC112 Spring 2011
Fundamentals of Computer Science
Practice Problems – Solutions

Note: Some of these problem are too long (time-wise) to include on a 50-minute exam. Also, there are too many problems here to complete in a 50-minute exam. But, they review the material, and help prepare you for the exam.

1. Write a C++ function to accept an array of characters (C-style character strings, terminated by the null character `'\0'`) and return the length of the string. *Hint:* Count the number of characters up to the character `'\0'`. The function header should be:

   ```
   int string_length( char * s )
   ```

   Try this exercise again, with the function header:

   ```
   int string_length( char s[] )
   ```

   Solution:

   Using a pointer:

   ```
   int string_length(char * s)
   {
      int count ;

      count = 0 ;
      while ( (*s) != '\0' ) {
         count++ ; s++ ;
      }
      return(count) ;
   }
   ```

   Using an array:

   ```
   int string_length(char s[])
   {
      int count ;

      count = 0 ;
      while ( s[count] != '\0' ) {
         count++ ;
      }
      return(count) ;
   }
   ```

2. Describe how to implement a 2-D array using a pointer to a pointer. Suppose the declaration of an array is:

   ```
   int m, n       ;
   double ** table ;
   ```

   Write a segment of code to allocate memory for a 2-D array with m rows and n columns.

Solution:

```
int i ;
table = new double * [ m ] ;
for ( i = 0 ; i < m ; i++ )  {
   table[i] = new double [ n ] ;
}
```

3. A *Hilbert matrix* is a matrix whose entries are given by:

$$H_{i,j} = \frac{1}{i+j+1}$$

where $i, j \in \{0, 1, 2, ..., n\}$. Write a segment of code to initialize a 7×7 Hilbert matrix.

Solution: A complete program including allocation and initialization is shown below:

```
#include <iostream>
using namespace std ;

int main()
{
   int i, j, n ;
   double ** hilbert ;

   n = 7 ;

   hilbert = new double * [ n ] ;     // Allocate memory
   for ( i = 0 ; i < n ; i++ )  {
       hilbert[i] = new double [ n ] ;
   }

   for ( i = 0 ; i < n ; i++ )  {    // Initialize
      for ( j = 0 ; j < n ; j++ )  {
          hilbert[i][j] = 1.0 / ( i + j + 1.0 ) ;
      }
   }
}
```

4. (**Slightly harder problem**) Write a C++ function to accept two arrays of characters (C-style character strings), `s` and `t`. Find the first occurrence of substring `t` in string `s`. Your function should return a (char) pointer to where the match occurs in string `s`. If no match is found, return the NULL pointer. The function header should be:

```
char * find_string( char *s, char * t )
```

Solution: Here is a complete program, including a sample output.

```
#include <iostream>
#include <cstdio>
using namespace std ;

char * find_string( char *s, char * t )
{
    char * u, * ui, * ti  ;
    bool loop, match ;

    u = s ;        //  u is the starting point looking for a match
    loop = true ;
    while (loop) {
       ui = u ;  ti = t ;
       match = true ;  //  Assume we have a match until proven otherwise.
       while ( (*ui != '\0') && (*ti != '\0') && match ) {
          match = (*ui) == (*ti) ;
          if (match) { ui++ ; ti++ ; }
       }
       if (match) {
          if (*ti == '\0') return(u) ; //  Match found starting at u.
          else return(NULL) ; // End of u string before end of t
       }
       else {  // match is false.  Increment u and try again.
          u++ ;
       }
    }
}

void print_result( char * ch_ptr, char * target )
{
   if ( ch_ptr == NULL )
      cout << "target '" << target << "' not found. " << endl ;
   else
      cout << "target '" << target << "' found:  '" << ch_ptr << "'" << endl ;
}

// ---------------------------------------------------------------
int main()
{
   char * p, * q ;
   char * source =  (char *) "The quick red fox jumped over" ;
   char * target1 =  (char *) "fox" ;
   char * target2 =  (char *) "box" ;

   p = find_string( source, target1 ) ;
   print_result( p, target1 ) ;
   q = find_string( source, target2 ) ;
   print_result( q, target2 ) ;
}
```

Here is a sample session:

```
atlas% g++ find_string.cc
atlas% a.out
target 'fox' found:  'fox jumped over'
target 'box' not found.
```

5. Write a C++ function to accept the name of a file containing binary integers, and compute the sum of the numbers in the file. The function header should be:

```
int file_sum( char * fname )
```

Solution: Here is a complete program.

```
#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std ;

int file_sum( char * fname )
{
   ifstream bf  ;
   bool loop    ;
   int k, total ;
   bf.open( fname, ios::binary | ios::in ) ;
   if ( bf.fail() ) {
       cerr << "Unable to open file '" << fname << "'" << endl ;
       exit(1) ;
   }

   loop = true ;
   total = 0   ;
   while (loop) {
      bf.read( (char *) &k, sizeof(int) ) ;
      loop = !bf.eof() ;
      if (loop) {
        total += k ;
      }
   }
   return(total) ;
}

int main()
{
   const char * fname = "binary_numbers" ;
   int t ;

   t = file_sum( (char *) fname ) ;
   cout << "file sum is " << t << endl ;
}
```

6. Write a C++ function to accept the name of a text file and count the number of sentences in that file. Assume every sentence ends with a period, and there are no other uses of the period character in the file. I.e., read the file one character at a time and count the number of periods. The function header should be:

```
int count_periods( char * fname )
```

Solution: Here is a complete program.

```
#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std ;

int count_periods( char * fname )
{
   ifstream tf  ;
   bool loop    ;
   int count    ;
   char ch      ;

   tf.open( fname, ios::in ) ;
   if ( tf.fail() ) {
       cerr << "Unable to open file '" << fname << "'" << endl ;
       exit(1) ;
   }

   loop = true ;
   count = 0   ;
   while (loop) {
      tf.get( ch ) ;
      loop = !tf.eof() ;
      if ( loop && (ch == '.') )  count++ ;
   }
   return(count) ;
}

int main()
{
   const char * fname = "text_file" ;
   int t ;

   t = count_periods( (char *) fname ) ;
   cout << "Number of period punctuation is " << t << endl ;
}
```

7. Give the declaration of a C++ structure (named `struct polynomial`). to implement a polynomial. You may assume that any polynomial that we will use is of degree less than 40. An example of a polynomial is given by:

$$p(x) = 2 + x - 3x^2 + 5x^3 + x^4 - 2x^5$$

Hint: A polynomial can be represented by a list of its coefficients.

Solution:

```
const int MAX_COEFFS = 40 ;

struct polynomial {
   int n ;
   double coefficients[MAX_COEFFS] ;
} ;
```

8. Using your declaration from problem 7, write a C++ function to evaluate the polynomial at x. The function header should be:

```
double polynomial_eval(struct polynomial p, double x)
```

Solution:

```
double polynomial_eval(struct polynomial p, double x)
{
   int k ;
   double x_to_k ;
   double r ;

   x_to_k = 1.0 ;
   r = 0 ;
   for ( k = 0 ; k < p.n ; k++ ) {
      r += p.coefficients[k] * x_to_k ;
      x_to_k *= x ;
   }
   return(r) ;
}
```

9. Suppose the following structure is used to implement an un-ordered list integers:

```
const int max_nums = 1024 ;
struct numlist {
    int n ;                 // n is the number of items on the list
    int nums[ max_nums ] ;
} ;
```

Write a C++ function to find a particular item on the list. Your function should return the position on the list where the item is found; return -1 if the item is not on the list. The function header should be:

```
int find_num( struct numlist & nu, int item )
```

Solution: Here is a complete program.

```
#include <iostream>
using namespace std ;

const int max_nums = 1024 ;
struct numlist {
   int n ;                  // n is the number of items on the list
   int nums[ max_nums ] ;
} ;

int find_num( struct numlist & nu, int item )
{
    int i ;
    bool found ;

    found = false ;
    i = 0 ;
    while ( ( i < nu.n ) && (!found) ) {   // Use linear search.
       found =  item == nu.nums[i] ;
       if ( !found ) i++ ;
    }
    if (found) return i ;
    else return -1      ;
}

void init_list( struct numlist & nu )
{
    const int m = 11 ;
    int a[m] = { 2, 11, 7, 13, 27, 18, 6, 5, 22, 4, 8 } ;

    for ( int i = 0 ; i < m ; i++ ) {  nu.nums[i] = a[i] ; }
    nu.n = m ;
}

// ---------------------------------------------------------------
int main()
{
    struct numlist alist ;
    int t = 5 ;
    int position   ;

    init_list( alist ) ;
    position = find_num( alist, t ) ;

    if ( position < 0 ) {
      cout << "Target " << t << " not found." << endl ;
    }
    else {
      cout << "Target " << t << " in position " << position << endl;
    }
}
```

```
--------------------------- Sample session --------------------------

atlas% g++ find.cc
atlas% a.out
Target 5 found in position 7
```

10. Use your result (the position number) from problem 9 to delete the item from the list. Write a C++ function to do this. The function header should be:

```
void delete_num( struct numlist & nu, int position )
```

Solution:

```
void delete_num( struct numlist & ali, int position )
{
   int j ;
   for ( j = (position+1) ; j < ali.n ; j++ ) {
      ali.nums[j-1] = ali.nums[j] ;
   }
   ali.n-- ;
}
```

11. Suppose the following structure is used to implement an alphabetized list of names:

```
const int max_names = 100 ;
struct namelist {
    int n ;                  // n is the number of items on the list
    char * the_names[ max_names ] ;
} ;
```

Write a C++ function to insert a new name in the correct position. The function header should be:

```
void insert_name( struct namelist & nlist, char * new_name )
```

Solution:

```
void insert_name( struct namelist & nlist, char * new_name )
{
    int j, u ;

    if ( nlist.n >= max_names ) {
       cerr << "No space left to insert." << endl ;
       exit(1) ;
    }

    j = nlist.n - 1 ;  // Start in the last position.
    u = -1 ;
    while ( ( u < 0 ) && ( j >= 0 ) )  {
       u = strcmp( new_name, nlist.the_names[j] ) ;
       if ( u < 0 ) {
           nlist.the_names[j+1] = nlist.the_names[j] ;
           j-- ;
       }
    }
    nlist.the_names[j+1] = strdup( new_name ) ;
    nlist.n++ ;
}
```

12. What does the following program print ? *Hint:* Draw a picture of memory (boxes labeled a, b, c, p, q, r). Trace the execution of this program line by line, updating your diagram after each line.

```
#include <iostream>
using namespace std ;
int main()
{
   int a, b, c ;
   int * p, * q, * r  ;

   a = 11 ; b = 13 ; c = 17 ;
   p = &a ; q = &b ; r = &c ;
   *q = a ;
   *r = b ;
   *p = c ;
   cout << a << " " << b << " " << c << endl ;
}
```

Solution:

```
11 11 11
```