CS1010E Programming Methodology
Semester 1 2016/2017

Week of 31 October – 4 October 2016
Tutorial 10 Suggested Answers
Character Strings and Pointers

1. You have been introduced to the four string functions: `strlen`, `strcpy`, `strcat` and `strcmp`. In this exercise, you will implement your own string functions. In each of the following parts, write a `main` function to test the correctness of your implementation and compare with the string library equivalent. You are not allowed to use any of the string functions from the string library; however, you may call your own functions.

   (a) Implement the `my_strlen(s)` function that returns the number of characters in string `s` excluding the termination null character.

   ```
   int my_strlen(const char *s);
   ```

   (b) Implement the `my_strcpy(s1,s2)` function that copies string `s2` to `s1`, including the terminating null character, stopping after the null character has been copied. String `s1` is returned.

   ```
   char *my_strcpy(char *s1, const char *s2);
   ```

   (c) Implement the `my_strcat(s1,s2)` function to append a copy of string `s2`, including the terminating null character, to the end of string `s1`. The initial character of `s2` overrides the null character at the end of `s1`. String `s1` is returned.

   ```
   char *my_strcat(char *s1, const char *s2);
   ```

   (d) Implement the `my_strcmp(s1,s2)` function that compares two strings character-by-character, according to the ASCII character ordering. The function returns an integer greater than, equal to, or less than 0, if the string `s1` is greater than, equal to, or less than the string `s2`.

   ```
   int my_strcmp(const char *s1, const char *s2);
   ```

```
#include <stdio.h>

int my_strlen(const char *s);
char *my_strcpy(char *s1, const char *s2);
char *my_strcat(char *s1, const char *s2);
int my_strcmp(const char *s1, const char *s2);

int main(void) {
   char word[40]="cs1010e";

   printf("%d\n", my_strlen(word));
   printf("%s\n", my_strcpy(word,"CS1010E"));
   printf("%s\n", my_strcat(word," is fun!"));
   printf("%d\n", my_strcmp(word,"CS1010E is fun!"));

   return 0;
}

/*
   my_strlen returns the number of characters in string s.
   Precondition: s must be null terminated
*/
int my_strlen(const char *s) {
   int i = 0;

   while (s[i] != '\0')
      i++;

   return i;
}

/*
   my_strcpy copies string s2 to s1.  String s1 is returned.
   Precondition: s2 must be null terminated
*/
char *my_strcpy(char *s1, const char *s2) {
   int i = 0;

   while (s2[i] != '\0') {
      s1[i] = s2[i];
      i++;
   }
   s1[i] = '\0';

   return s1;
}
```

```
/*
   my_strcat concatenates string s2 to the end of s1.
   String s1 is returned.
   Precondition: s1 and s2 must be null terminated
*/
char *my_strcat(char *s1, const char *s2) {
   int n = my_strlen(s1);

   my_strcpy(s1 + n, s2);

   return s1;
}

/*
   my_strcmp(s1,s2) compares s1 and s2 lexicographically.
   Returns a negative value if s1 precedes s2, a positive value if
   s2 precedes s1, and zero if s1 is the same as s2.
   Precondition: s1 and s2 must be null terminated
*/
int my_strcmp(const char *s1, const char *s2) {
   int i = 0;

   while (s1[i] != '\0' && s2[i] != '\0') {
      if (s1[i] != s2[i]) {
         return s1[i] - s2[i];
      }
      i++;
   }

   return s1[i] - s2[i];
}
```

2. Christmas comes early for cs1010e when we get to sing our favourite Christmas carol "The Twelve Days of Christmas". The song enumerates a series of gifts given on each of the twelve days of Christmas in the following order:

   - a partridge in a pear tree
   - two turtle doves
   - three French hens
   - four calling birds
   - five golden rings
   - six geese a laying
   - seven swans a swimming
   - eight maids a milking
   - nine ladies dancing
   - ten lords a leaping
   - eleven pipers piping
   - twelve drummers drumming

   The carol is in the form of a cumulative song with lyrics as follows:

```
On the first day of Christmas,
my true love sent to me,
a partridge in a pear tree.

On the second day of Christmas,
my true love sent to me,
two turtle doves,
and a partridge in a pear tree.

On the third day of Christmas,
my true love sent to me,
three French hens,
two turtle doves,
and a partridge in a pear tree.

On the fourth day of Christmas,
my true love sent to me,
four calling birds,
three French hens,
two turtle doves,
and a partridge in a pear tree.

On the fifth day of Christmas,
my true love sent to me,
five golden rings,
four calling birds,
three French hens,
two turtle doves,
and a partridge in a pear tree.

On the sixth day of Christmas,
my true love sent to me,
six geese a laying,
five golden rings,
four calling birds,
three French hens,
two turtle doves,
and a partridge in a pear tree.

On the seventh day of Christmas,
my true love sent to me,
seven swans a swimming,
six geese a laying,
five golden rings,
four calling birds,
three French hens,
two turtle doves,
and a partridge in a pear tree.

On the eighth day of Christmas,
my true love sent to me,
eight maids a milking,
seven swans a swimming,
six geese a laying,
five golden rings,
four calling birds,
three French hens,
two turtle doves,
and a partridge in a pear tree.

On the ninth day of Christmas,
my true love sent to me,
nine ladies dancing,
eight maids a milking,
seven swans a swimming,
six geese a laying,
five golden rings,
four calling birds,
three French hens,
two turtle doves,
and a partridge in a pear tree.

On the tenth day of Christmas,
my true love sent to me,
ten lords a leaping,
nine ladies dancing,
eight maids a milking,
seven swans a swimming,
six geese a laying,
five golden rings,
four calling birds,
three French hens,
two turtle doves,
and a partridge in a pear tree.

On the eleventh day of Christmas,
my true love sent to me,
eleven pipers piping,
ten lords a leaping,
nine ladies dancing,
eight maids a milking,
seven swans a swimming,
six geese a laying,
five golden rings,
four calling birds,
three French hens,
two turtle doves,
and a partridge in a pear tree.

On the twelfth day of Christmas,
my true love sent to me,
twelve drummers drumming,
eleven pipers piping,
ten lords a leaping,
nine ladies dancing,
eight maids a milking,
seven swans a swimming,
six geese a laying,
five golden rings,
four calling birds,
three French hens,
two turtle doves,
and a partridge in a pear tree.
```

The carol can be generated cumulatively with a program using two arrays of strings representing the ordinal numbers and the gifts. Since each string is a 1D array of characters, it follows that an array of strings is a 2D array of characters. For example,

```
char ordinal[12][9]={"first","second","third","fourth","fifth","sixth",
                     "seventh","eighth","ninth","tenth","eleventh","twelfth"};
```

(a) Write a function `generateSong` that generates the lyrics of the song from day 1 to day n in the string `str`. The function returns the address of the string `str`.

```
char *generateSong(char *str, int n);
```

Declare the arrays of strings representing the ordinal numbers and gifts within the `generateSong` function. Assume the entire song will not exceed 3000 characters.

(b) Write a function `countWords` to count the number of words in the string `lyrics`.

```
int countWords(char *str);
```

(c) Write a `main` function to request the user for the number of days n (from 1 to 12), and generate the song from day 1 to day n. A sample run is given as follows. User input is underlined.

```
Enter number of days: 5

On the first day of Christmas,
my true love sent to me,
a partridge in a pear tree.

On the second day of Christmas,
my true love sent to me,
two turtle doves,
and a partridge in a pear tree.

On the third day of Christmas,
my true love sent to me,
three French hens,
two turtle doves,
and a partridge in a pear tree.

On the fourth day of Christmas,
my true love sent to me,
four calling birds,
three French hens,
two turtle doves,
and a partridge in a pear tree.

On the fifth day of Christmas,
my true love sent to me,
five golden rings,
four calling birds,
three French hens,
two turtle doves,
and a partridge in a pear tree.

The number of words is 124
```

```
#include <stdio.h>
#include <stdbool.h>
#include <string.h>

char *generateSong(char *lyrics, int n);
int countWords(char *lyrics);
bool isAlpha(char c);

int main(void) {
   int n;
   char lyrics[3000]="";

   printf("Enter number of days: ");
   scanf("%d", &n);
   printf("%s", generateSong(lyrics,n));
   printf("The number of words is %d\n", countWords(lyrics));

   return 0;
}

/*
   generateSong constructs the lyrics of "The Twelve Days of Christmas"
   for the first n days.

   Precondition: none
*/
char *generateSong(char *str, int n) {
   int i, j;
   char order[13][10] = {"","first","second","third",
                         "fourth","fifth","sixth",
                         "seventh","eighth","ninth",
                         "tenth","eleventh","twelfth"};

   char gift[13][30] = {"","a partridge in a pear tree.",
                         "two turtle doves,",
                         "three French hens,",
                         "four calling birds,",
                         "five golden rings,",
                         "six geese a laying,",
                         "seven swans a swimming,",
                         "eight maids a milking,",
                         "nine ladies dancing,",
                         "ten lords a leaping,",
                         "eleven pipers piping,",
                         "twelve drummers drumming,"};
```

```
   for (i = 1; i <= n; i++) {
      strcat(str, "On the ");
      strcat(str, order[i]);
      strcat(str, " day of Christmas,\nmy true love sent to me,\n");

      for (j = i; j >= 1; j--) {
         if (j == 1 && i > 1) {
            strcat(str, "and ");
         }
         strcat(str, gift[j]);
         strcat(str, "\n");
      }
      strcat(str,"\n");
   }

   return str;
}

/*
   isAlpha returns true if c is an alphabet, false otherwise.

   Precondition: none.
*/
bool isAlpha(char c) {
   return (c >= 'A' && c <= 'Z') ||
          (c >= 'a' && c <= 'z');
}

/*
   countWords returns the number of words in the string str.

   Precondition: str must be null terminated
*/
int countWords(char *str) {
   int count = 0;

   while (*str != '\0') {
      if (isAlpha(*str) && !isAlpha(*(str+1))) {
         count++;
      }
      str++;
   }
   return count;
}
```

3. Two words are anagrams if one word can be formed from the other word by rearranging the letters. For example "listen" and "silent" are anagrams. In this question, assume that both words are made up of lowercase characters. No assumption can be made on the maximum length of the word.

   (a) Write a function `isAnagram1` that takes as argument two strings `str1` and `str2` and returns `true` if they are anagrams, or `false` otherwise. Solve the problem by sorting the strings and comparing them. You may modify `str1` and `str2`.

   ```
   bool isAnagram1(char *str1, char *str2);
   ```

   (b) Write a function `isAnagram2` that takes as argument two strings `str1` and `str2` and returns `true` if they are anagrams, or `false` otherwise. Use the strategy of search and mark, i.e. for every character of `str1`, find the corresponding letter in `str2` and mark it by replacing with a non-lowercase character so that it would not be checked again. You may only modify the string `str2`.

   ```
   bool isAnagram2(const char *str1, char *str2);
   ```

   (c) Write a function `isAnagram3` that takes as argument two strings `str1` and `str2` and returns `true` if they are anagrams, or `false` otherwise. Use the strategy of frequency counting, i.e. counting the occurrences of each letter. You are **not allowed** to modify the strings `str1` and `str2`.

   ```
   bool isAnagram3(const char *str1, const char *str2);
   ```

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>

bool isAnagram1(char *str1, char *str2);
void sort(char str[], int n);
bool isAnagram2(const char *str1, char *str2);
int search(char str[], int n, char c);
bool isAnagram3(const char *str1, const char *str2);

int main(void) {
   char str1[40], str2[40];

   printf("Enter two words seperated by space: ");
   scanf("%s %s", str1, str2);
   printf("%s and %s ", str1, str2);
   if (isAnagram1(str1,str2)) {  /* try with isAnagram2 or isAnagram3 */
      printf("are anagrams.\n");
   } else {
      printf("are not anagrams.\n");
   }
   return 0;
}
```

```
/*
   isAnagram1 test anagrams by sorting both str1 and str2.

   Precondition: str1 and str2 are null terminated.
*/
bool isAnagram1(char *str1, char *str2) {
   int n1=(int)strlen(str1), n2=(int)strlen(str2);

   if (n1 != n2) { /* perform trivial test based on lengths */
      return false;
   }
   sort(str1,n1);
   sort(str2,n2);
   return strcmp(str1,str2) == 0;
}

void sort(char str[], int n) {
   int i, j, minIndex;
   char temp;

   for (i = 0; i < n-1; i++) {
      minIndex = i;
      for (j = i+1; j < n; j++) {
         if (str[j] < str[minIndex]) {
            minIndex = j;
         }
      }
      temp = str[i];
      str[i] = str[minIndex];
      str[minIndex] = temp;
   }
   return;
}


/*
   isAnagram2 tests anagrams by searching and marking str2.

   Precondition: str1 and str2 are null terminated.
*/
bool isAnagram2(const char *str1, char *str2) {
   int i, j, n=(int)strlen(str1);
   bool anag=true;

   if (n != (int)strlen(str2)) {
      return false;
   }
   i = 0;
```

```
   while (i < n && anag) {
      j = search(str2,n,str1[i]);
      if (j == -1) {
         anag = false;
      } else {
         str2[j] = '.';  /* mark with '.' */
      }
      i++;
   }
   return anag;
}

int search(char str[], int n, char c) {
   int k=0;

   while (k < n && str[k] != c) {
      k++;
   }
   if (k < n) {
      return k;
   } else {
      return -1;
   }
}

/*
   isAnagram3 tests anagrams by table-lookup and frequency counting.

   Precondition: str1 and str2 are null terminated.
*/
bool isAnagram3(const char *str1, const char *str2) {
   int freq[26]={0}, i, n=(int)strlen(str1);
   bool anag=true;

   if (n != (int)strlen(str2)) {
      return false;
   }
   for (i = 0; i < n; i++) {
      (freq[str1[i]-'a'])++;
      (freq[str2[i]-'a'])--;
   }
   for (i = 0; i < 26 && anag; i++) {
      if (freq[i] != 0) {
         anag = false;
      }
   }
   return anag;
}
```

4. Re-implement the `binarySearch` function in lecture #9 by using pointers to reference the left, right and middle elements. Complete the function using the program fragment below.

```
int binarySearch(int x[], int n, int v) {
   int *left, *right, *mid;

   left = x;
   right = x + n - 1;
   /* complete the function here */
}
```

Test the `binarySearch` function using the `main` function given below.

```
int main(void) {
   int x[10] = {7, 13, 20, 38, 44, 52, 88, 89, 90, 92}, val, index;

   printf("Enter number to find: ");
   scanf("%d", &val);
   index = binarySearch(x, 10, val);
   printf("%d is found at index %d\n", val, index);

   return 0;
}
```

```
#include <stdio.h>

int binarySearch(int x[], int n, int v);

int main(void) {
   int x[10] = {7, 13, 20, 38, 44, 52, 88, 89, 90, 92}, val, index;

   printf("Enter number to find: ");
   scanf("%d", &val);
   index = binarySearch(x, 10, val);
   printf("%d is found at index %d\n", val, index);

   return 0;
}

/*
   binarySearch returns the index of x where v is found.
   Returns -1 in the case of an unsuccessful search.

   Precondition: none.
*/
int binarySearch(int x[], int n, int v) {
   int *left, *right, *mid;            /* declared as pointers */

   left = x; right = x + n - 1;        /* point to ends of array */
   while (left <= right) {             /* compare addresses */
      mid = left + (right - left) / 2; /* alternative way to find middle */
      if (*mid == v) {
         return mid - x;               /* return index */
      } else if (*mid > v) {
         right = mid - 1;
      } else {
         left = mid + 1;
      }
   }
   return -1;
}
```