CptS 121 - Program Design and Development

WASHINGTON STATE
UNIVERSITY
World Class. Face to Face.

Programming Assignment 8: C Interview Questions

Assigned: Wednesday, April 23, 2014
Due: Friday, May 2, 2014, by midnight

I. Learner Objectives:

At the conclusion of this programming assignment, participants should be able to:

- Apply and implement all your problems solving and C skills developed this semester!
- Apply and implement pointers in C
- Manipulate and split arrays and strings
- Apply and implement recursive functions

II. Prerequisites:

Before starting this programming assignment, participants should be able to:

- Analyze a basic set of requirements and apply top-down design principles for a problem
- Apply repetition structures within an algorithm
- Construct while (), for (), or do-while () loops in C
- Compose C programs consisting of sequential, conditional, and iterative statements
- Eliminate redundancy within a program by applying loops and functions
- Create structure charts for a given problem
- Open and close files
- Read, write to, and update files
- Apply standard library functions: fopen (), fclose (), fscanf (), and fprintf ()
- Compose decision statements ("if" conditional statements)
- Create and utilize compound conditions

III. Overview & Requirements:

For this assignment you will be required to write functions which solve each of the following problems. You must place all of your functions in one project. If you use any code that you find online, you must reference it in comments.

1. (10 pts) Write a function called `my_str_n_cpy()` that accepts *pointer* to a *destination* character array and a *pointer* to a *source* character array (which is assumed to be a string) and returns the *pointer* to the *destination* character array. This function needs to copy at most *n* characters, character by character, from the source character array to the destination character array. If a null character is encountered before n characters have been encountered, copying must stop. You may NOT use any functions found in <string.h> to solve this problem! Note: you may use array or pointer notation in this function.

2. (10 pts) Recall Binary Search:

Input: a list of n *sorted* integer values and a target value
Output: True if target value exists in list and location of target value, false otherwise

```
    Method:
        Set left to 1 and right to n
```

```
Set found to false
Set targetindex to -1
While found is false and left is less than or equal to right
    Set mid to midpoint between left and right
    If target = item at mid then set found to true and set targetindex to mid
    If target < item then set right to mid – 1
    If target  > item then set to left to mid + 1
Return the targetindex
```

Write a C function called `binary_search()`.

3. (15 pts) Write a function called `bubble_sort()` that accepts an *array* of *pointers* to *strings* and the *number* of strings as arguments, and returns nothing. The function sorts the strings according to the following algorithm:

```
1.  set the marker U for the unsorted section at the end of the list (U is an integer index value)
2.  while the unsorted section has more than one element do steps 3 through 7
3.     set the current element marker C at the second element of the list (C is an integer index value)
4.     while C has not passed U do steps 5 and 6
5.           if the item at position C is less than the item to its left then
              exchange these two items
6.           move C to the right one position
7.     move U left one position
8.  stop
```

Your implementation for this function may NOT use `strcpy()`. You may only exchange or swap pointers, but NOT actually make *copies* of the strings!

4. (15 pts) Write a recursive function called `is_palindrome()` that accepts a pointer to a *string* and its *length*, and *recursively* determines if the string is a palindrome. The function must return 1 for a palindrome, 0 otherwise. A palindrome is a sequence of symbols that may be interpreted the same forward and backward. For example, "race car". Note: whitespace should be ignored in your solution.

5. (15 pts) Write a recursive function called `sum_primes()` that accepts an *unsigned* integer, `n`, as an argument, and returns the *sum* of all primes from 2 to n. You must use recursion to solve this problem!

6. (20 pts) Write a function called `maximum_occurences()` that accepts a *pointer* to a *string* (consisting of alphanumeric and whitespace characters only), a *pointer* to an *array* of *struct occurrences*, a *pointer* to an *integer*, and a *pointer* to a *character* as arguments. The structure is defined as follows:

```
typedef struct occurrences
{
        int num_occurrences;
        double frequency;
} Occurrences;
```

The function determines the frequency of each character found in the array. The frequency is defined as: number of one character symbol / total number of characters. The function should use the second array argument (of struct occurrences) to keep track of the frequency of each character. Also, it must return, through the pointers, the maximum number of occurrences of any one character and the corresponding character for which the maximum represents. Thus, for a string such as "test string", 't' occurs 3 times, which is the maximum occurrences for any one character in the string.

7. (20 pts) Write a function called `sum_smallest_sequence()` that accepts an array of *signed* integers and the *number* of items in the array as arguments, and returns the sum of the smallest sequence of

numbers in the array. A *sequence* is defined as a single item or multiple items that are in adjacent memory locations.

8. (BONUS – 20 pts) Write a function called `max_consecutive_integers()` that accepts a two-dimensional array of *signed* integers, the number of *rows*, the number of *columns* as input parameters, and two *pointers* as output parameters (one of these pointers is actually a pointer to a pointer, i.e. two stars!). The function finds the maximum consecutive sequence of one integer. The first pointer stores the address the start of the maximum consecutive sequence of the same integer. The second indirectly stores the number the same consecutive integers in a row. These sequences may wrap from one row to the next. For example ([$xxxx] denotes address value):

Row/Column	0	1	2	3	4
0	-5 [$1000]	6 [$1004]	0 [$1008]	2 [$1012]	2 [$1016]
1	2 [$1020]	2 [$1024]	2 [$1028]	9 [$1032]	3 [$1036]
2	3 [$1040]	3 [$1044]	2 [$1048]	1 [$1052]	-8 [$1056]
3	7 [$1060]	-2 [$1064]	6 [$1068]	0 [$1072]	4 [$1076]

The function should store the address of row 0, column 3 ($1012) via the first pointer, and 5 (2, 2, 2, 2, 2) *indirectly* via the second pointer.

IV. Submitting Assignments:

1. Using the Angel tool https://lms.wsu.edu submit your assignment to your TA. You will "drop" your solution into the provided "Homework Submissions" Drop Box under the "Lessons" tab. You must upload your solutions as <your last name>_pa8.zip by the due date and time.
2. Your .zip file should contain project workspace. Within the project you must have at least one header file (a .h file), two C source files (which must be .c files), and project workspace. Delete the debug folders before you zip the project folder.
3. Your project must build properly. The most points an assignment can receive if it does not build properly is 75 out of 115.

V. Grading Guidelines:

This assignment is worth 115 points. Your assignment will be evaluated based on a successful compilation and adherence to the program requirements. We will grade according to the following criteria:

- 105 pts for adherence to the instructions stated above (see the individual points above)
- 10 pts for appropriate top-down design of functions and good style
- BONUS: Up to 20 pts for question 8