CS211 – Programming Practicum Spring 2020

Programming Project 1

Due: Wednesday, 2/5/20 at 11:59 pm

For your first project, write **a C program (not a C++ program!)** that will read in a given list of non-negative integers and a target integer and checks if there exist two integers in the list that sum up to the target integer.

Example:

*List***:** 31, 5, 8, 28, 15, 21, 11, 2
Target: 26 Yes!, 44 No!

Your C program will contain the following:

- Write a function that will make a copy of the values from one array to another array. Suggested prototype:
  void makeArrayCopy (int fromArray[], int toArray[], int size);

- Write your own function that will sort an array in ascending order. You may use whatever sorting algorithm you wish. Suggested prototype:
  void myFavoriteSort (int arr[], int size);

- Write your own function that will find whether there exist two integers that sum to the target integer. **The function is to "return" three values.** First, return "1" if the integers were found, return "-1" if your search was not successful. If you find two integers which add up to the target value, you should return their respective index position inside the array.
  Suggested prototype:
  int TwoSumFunction (int arr[], int size, int target, int* index1, int* index2);

Inside TwoSumFunction:

- Pass the sorted array to the function. Set two pointers **at the beginning and the end of the array**, then start moving the pointers inward while checking their sum. If it's exactly the "target", then we are done, and you can return 1. If it exceeds the "target" value, then any sum using the larger element is too large, so move the pointer corresponding to that element inward. If the sum is less than "target" value, then any sum using the lower element is too small, so move the pointer corresponding to that element inwards. If you are done with scanning the array and cannot find any two elements that sum up to "target" value, return -1.

Inside of main:

- Read in integer input from **standard input** and store these values into a dynamic array. This array is to grow in size if the array is full. The values will have a "terminal value" of -999. So, you read in these values in a loop that stops when the value of -999 is read in. The use of informative prompts is required for full credit. You may not assume how many numeric values are given on each line, nor the total number of values contained in

the input. **The use of a scanf() with the following form is expected** to read in the values:

scanf ("%d", &val);

- make a copy of the integer array using the arrayCopy() function described above
- sort the copy array (using the myFavoriteSort() function described above)
- read in integer input from standard input (again, the use of scanf() is expected) and for each of the values read in perform the TwoSum evaluation. Using the information returned/sent back from the search functions, **print out from main():**

    1. The target value,
    2. Whether the Two Sum evaluation was successful or not
    3. Locations of the elements in the array which make up the sum.

    The above information MAY NOT be printed out in TwoSum Function. The function MUST RETURN/SEND BACK the information to be printed by the main function. Not doing this will SEVERELY LOWER your score on this project.

    Repeat reading in integer values and searching the array until the terminal value of -999 is read in. The use of informative prompts AND descriptive result output is required for full credit. Again, scanf() is expected to read the input.

You may not assume the input will be less than any set size. Thus you will need to dynamically allocated space for the array.

Dynamic Array Allocation

Dynamic Array Allocation allows the space in an array to change during the course of the execution of a program. In C, this requires the use of the malloc() function. To dynamically allocate space for 100 integers, the malloc() code would be as follows:

```
int *darr;
int allocated = 100;
darr = (int *) malloc (allocated * sizeof(int) );
```

This array can only hold 100 integers and is not really dynamic. To make it truly dynamic, we need to grow the array when we try to put more values into it than can be held by its current size. The following code will double the size of the array.

```
int *temp = darr;
darr = (int *) malloc ( allocated * 2 * sizeof(int) );
int i;
for ( i = 0 ; i < allocated ; i++)
        darr[i] = temp[i];
free (temp);
allocated = allocated * 2;
```

CS211 – Programming Practicum Spring 2020

Running your C Program (not a C++ program)

Make sure your program runs properly when compiled using gcc on the bert.cs.uic.edu machine! (Do not use g++ or any other C++ compiler with this program.)

Programming Style

Make sure your program is written in good programming style. This includes but is not limited to:

- In-line commenting
- Function header commenting
- File header commenting
- Meaningful and description variable names
- Proper use of indentation
- Proper use of blank lines
- Use of Functions

Program Submission

You are to submit the program via the proper Assignments link on Gradescope. If you create your program in multiple files (**which you should not need to do for this project**), you should create a zip file containing all your files and then submit the zip file on Gradescope. You should name your files with your net-id and project name. For example:

The file should be named as netidproj1.c
The zip file (**if needed**) should be named as netidproj1.zip

Creating a zip file (if needed)

1. Go the directory in which you have saved the files.
2. Select the multiple files and right click.
3. There should be a menu option to create a zip file. If you are using Mac/Linux you will see a menu option **Compress....** If you are using Windows you might need to install winzip to create a zip file

Suggestion: Using Redirection of Input and Output to help Test Your Program

To help test your program, the use of redirection of standard input from a text file is a good idea for this project. Redirection is done at the command line using the less than and greater than signs. Redirection of both input and output can be done; however, for this project, you may only want to use redirection of input.

- Assume you have a text file that is properly formatted to contain the input as someone would type it in for the input called: **proj1input.txt**
- Also the executable for this project is in a file in the current directory called: **a.out**
- To run the project so that it reads the input from this text file instead of standard input using redirection of input, you would type the following on command line:

  ./a.out < proj1input.txt

- To store the output sent to standard output to a file called **outfile.txt** using redirection (the input is still being read from standard input), type:

./a.out > outfile.txt

- To redirect both standard input and standard output , type:
  ./a.out < proj1input.txt > outfile.txt

Note that the code inside of your program will still read from standard input. Redirection is information given to the Operating System at the command prompt. The Operating System then "redirects" a standard input read operation away from the keyboard and to the specified file while the program is running.