Programming Project 1

Due: Monday, 1/25/16 at 11:59 pm

Write a C program that will contain the following:

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

- Write your own function that will perform a linear search on the unsorted array. The function is to "return" two values. The first should be the position in the array where the value was found or -1 if the value was not found. The second is the number of comparisons needed to determine if/where the value is located in the array. Suggested prototype:
  int linSearch (int arr[], int size, int target, int* numComparisons);

- Write your own function that will perform a binary search on the sorted array. The function is to "return" two values. The first should be the position in the array where the value was found or -1 if the value was not found. The second is the number of comparisons needed to determine if/where the value is located in the array. Suggested prototype:
  int binSearch (int arr[], int size, int target, int* numComparisons);

Inside of main:

- read in integer input from **standard input** and store these values into an array. 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. The use of a scanf() with the following form is suggested to read in the values:
  scanf ("%d", &val);

- make a copy of the integer array using the arrayCopy() function described above

- sort one of the arrays (using the sort() function described above), leave the other array unsorted

Inside of main (continued):

- read in integer input from standard input and for each of these values search for the value using both search functions described above (i.e. perform a linear search on the value in the unsorted array and perform a binary search on the value in the sorted array). Using the information returned/sent back from the search functions, print out:

    1. Whether the value was found or not found,
    2. The number of comparisons needed to determine whether the value exists or not in each algorithm,
    3. And the location in the array where the number is located (if the value does exist in the array).

    Repeat reading in integer values and searching the arrays until the terminal value of -999 is read in. The use of informative prompts AND descriptive result output is required for full credit.

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

This array can only hold 100 integers and it 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;
temp = (int *) malloc ( size * 2 * sizeof(int) );
int i;
for ( i = 0 ; i < size ; i++)
        temp[i] = darr[i];
free (darr);
darr = temp;
size = size * 2;
```

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**
- Assume the executable for this project is in a file called: a.exe
- 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.exe < 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.exe > outfile.txt

- To redirect both standard input and standard output , type:
  a.exe < 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.

Program Submission

You are to submit the program via the proper Assignments link in Blackboard. 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 blackboard. You should name your files with your net-id and project name. For example if your net-id is ptroy1, then:

The file should be named as ptroy1proj1.c
The zip file (**if needed**) should be named as ptroy1proj1.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. In case of Mac/Linux you will see a menu option **Compress....** In case of Windows you might need to install winzip to create a zip file