CS 1037a – Computer Fundamentals II
Assignment 2

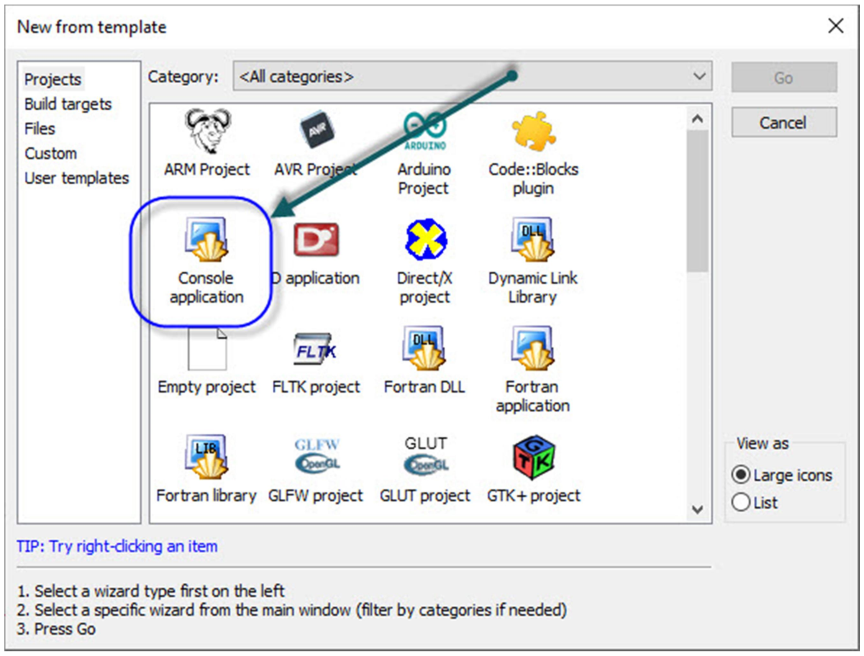
Instructor: L. Magguilli
Due Date: October 11, 2019, 6:00pm
Total marks: 100
Percentage of final mark: 3%
Accepted up to ONE (1) day late.

The purpose of this assignment is to provide the student with experience with arrays. This assignment will further provide experience using the basic code control structures covered in class.

PREPERATION:
Create a new project in Code::Blocks.

File->New->Project.

1.) Select the <Console Application> option from the [New from Template] screen.
2.) Select the C (NOT the C++) language
3.) Name the Project Title: *YourAccountName*_Asn02 (using your actual user name).
4.) Use the default compiler by clicking on the <Finish> button.



TIP: Try right-clicking an item

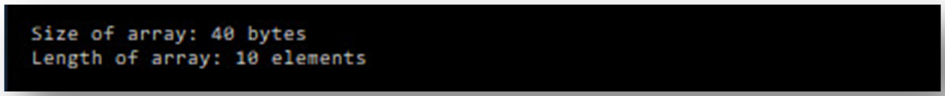
1. Select a wizard type first on the left
2. Select a specific wizard from the main window (filter by categories if needed)
3. Press Go

Under [Sources] in the Project window of Code::Blocks, open the main.c file.

Again, this assignment will be contained within a single main.c program.
The array **MUST** be initialized with the following values in the following order:

```
int array[] = {23,43,65,23,12,7,34,76,28,64};
```

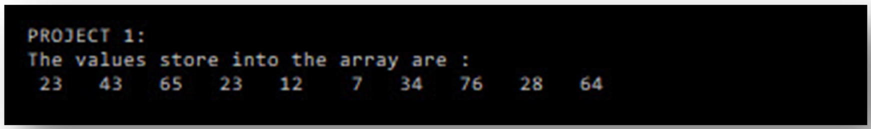
To initialize the assignment, compute the number of elements in the array and print out the size.
Your output should look like:



```
Size of array: 40 bytes
Length of array: 10 elements
```

Project 1: Print out the array
Simply print the array out exactly as it is stored.

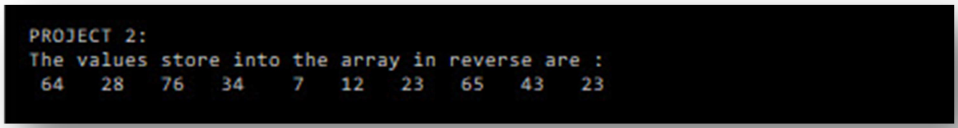
Your output should look like (five spaces between values):



```
PROJECT 1:
The values store into the array are :
 23    43    65    23    12     7    34    76    28    64
```

Project 2: Print the array in reverse order
Print the array out as it is stored, but in reverse order (last element printed first, second last, next, etc.
note: Do NOT change the actual array, just print it out in reverse.

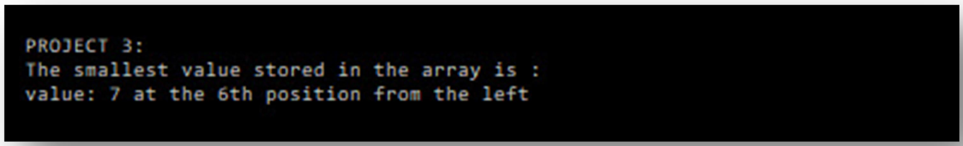
Your output should look like:



```
PROJECT 2:
The values store into the array in reverse are :
 64    28    76    34     7    12    23    65    43    23
```

Project 3: Find the smallest value currently stored in the array
Traverse the entire array and find the lowest of all the values in the array.
Print out the value and the actual position in the array the value was found
note: NOT the index of the value, but where it is in the array. (i.e. 28 is in the 2nd position).

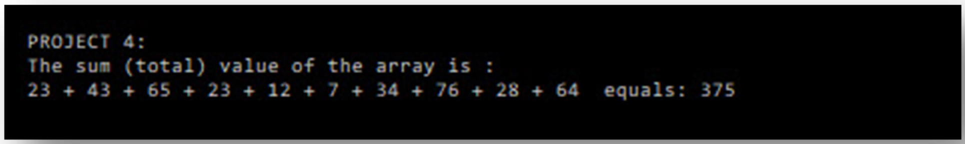
Your output should look like:



```
PROJECT 3:
The smallest value stored in the array is :
value: 7 at the 6th position from the left
```

Project 4: Add all the value in the array to compute the total (sum) of all the values
Print the equation and the result all on one line (including the addition symbol).
note: remember the last value does NOT have a 'plus' (+) sign after.

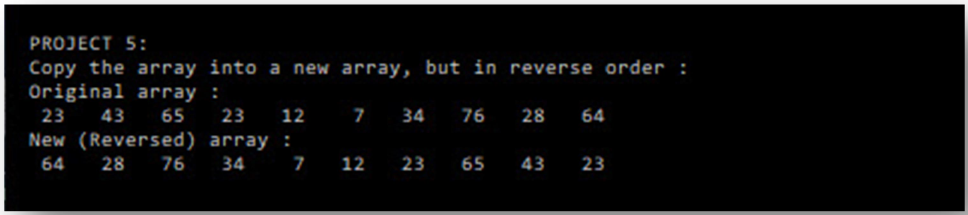
Your output should look like:



```
PROJECT 4:
The sum (total) value of the array is :
23 + 43 + 65 + 23 + 12 + 7 + 34 + 76 + 28 + 64  equals: 375
```

Project 5: Copy the original array into a new array, but in reverse order
Declare (but do not initially define) a new array.
Write code that will copy the values from the original array into the new array,
BUT in reverse order.
So, the last element of the original array is the first element of the new array. They second last element of the original array is now the second element of the new array, etc.

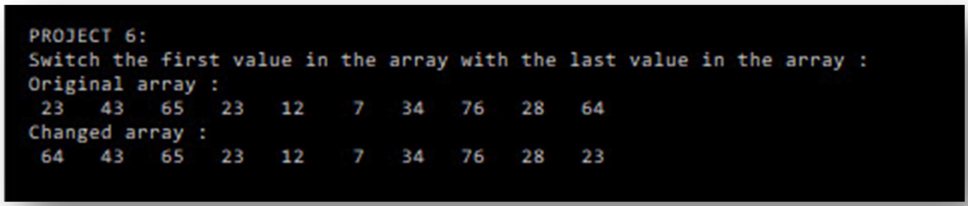
Your output should look like:



```
PROJECT 5:
Copy the array into a new array, but in reverse order :
Original array :
 23   43   65   23   12    7   34   76   28   64
New (Reversed) array :
 64   28   76   34    7   12   23   65   43   23
```

Project 6: Switch just the First and the Last elements of the array
Switch the value from the first position on the array with the value in the last position. The rest of the array remains unchanged.

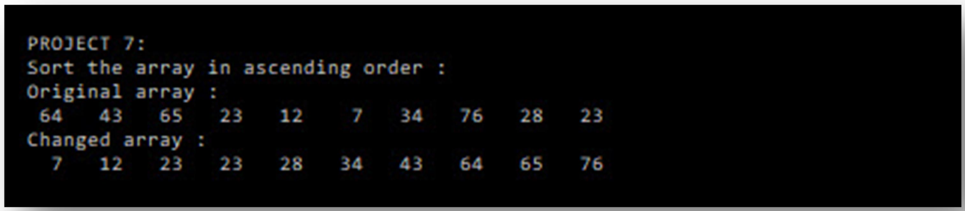
Your output should look like:



```
PROJECT 6:
Switch the first value in the array with the last value in the array :
Original array :
 23   43   65   23   12    7   34   76   28   64
Changed array :
 64   43   65   23   12    7   34   76   28   23
```

Project 7: Sort the array into ascending (lowest to highest) order
Using the concepts from the previous projects above, change the original array so the elements are now in ascending order.

Your output should look like:

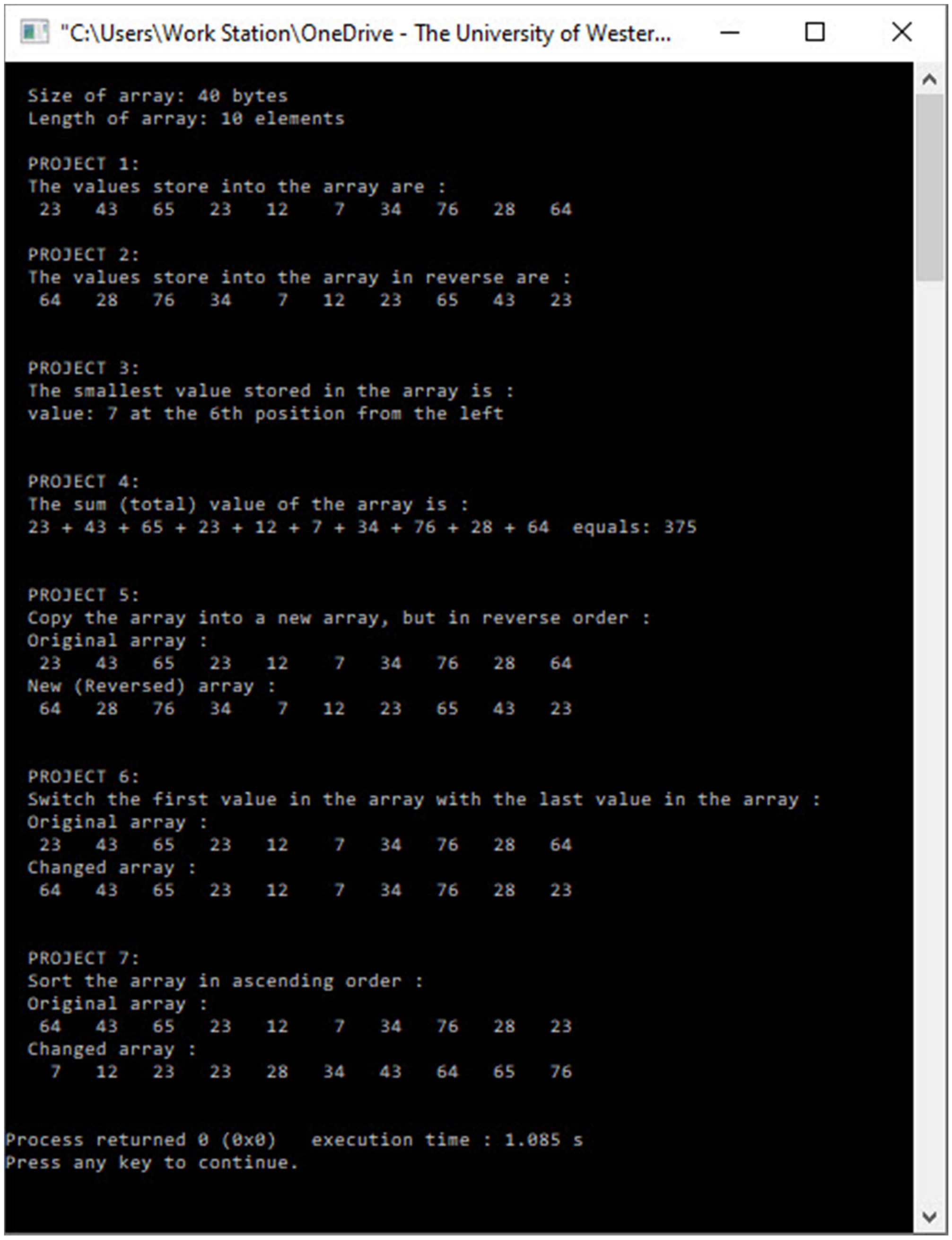


```
PROJECT 7:
Sort the array in ascending order :
Original array :
 64   43   65   23   12    7   34   76   28   23
Changed array :
  7   12   23   23   28   34   43   64   65   76
```

note: try to write this code without looking up the solution on the inter-webs.

These seven (7) projects are again to be completed in a single main.c and will execute one after the other.

The output from this single main program will look like:


"C:\Users\Work Station\OneDrive - The University of Wester...

```
 Size of array: 40 bytes
 Length of array: 10 elements

 PROJECT 1:
 The values store into the array are :
  23   43   65   23   12    7   34   76   28   64

 PROJECT 2:
 The values store into the array in reverse are :
  64   28   76   34    7   12   23   65   43   23


 PROJECT 3:
 The smallest value stored in the array is :
 value: 7 at the 6th position from the left


 PROJECT 4:
 The sum (total) value of the array is :
 23 + 43 + 65 + 23 + 12 + 7 + 34 + 76 + 28 + 64  equals: 375


 PROJECT 5:
 Copy the array into a new array, but in reverse order :
 Original array :
  23   43   65   23   12    7   34   76   28   64
 New (Reversed) array :
  64   28   76   34    7   12   23   65   43   23


 PROJECT 6:
 Switch the first value in the array with the last value in the array :
 Original array :
  23   43   65   23   12    7   34   76   28   64
 Changed array :
  64   43   65   23   12    7   34   76   28   23


 PROJECT 7:
 Sort the array in ascending order :
 Original array :
  64   43   65   23   12    7   34   76   28   23
 Changed array :
   7   12   23   23   28   34   43   64   65   76


Process returned 0 (0x0)   execution time : 1.085 s
Press any key to continue.
```

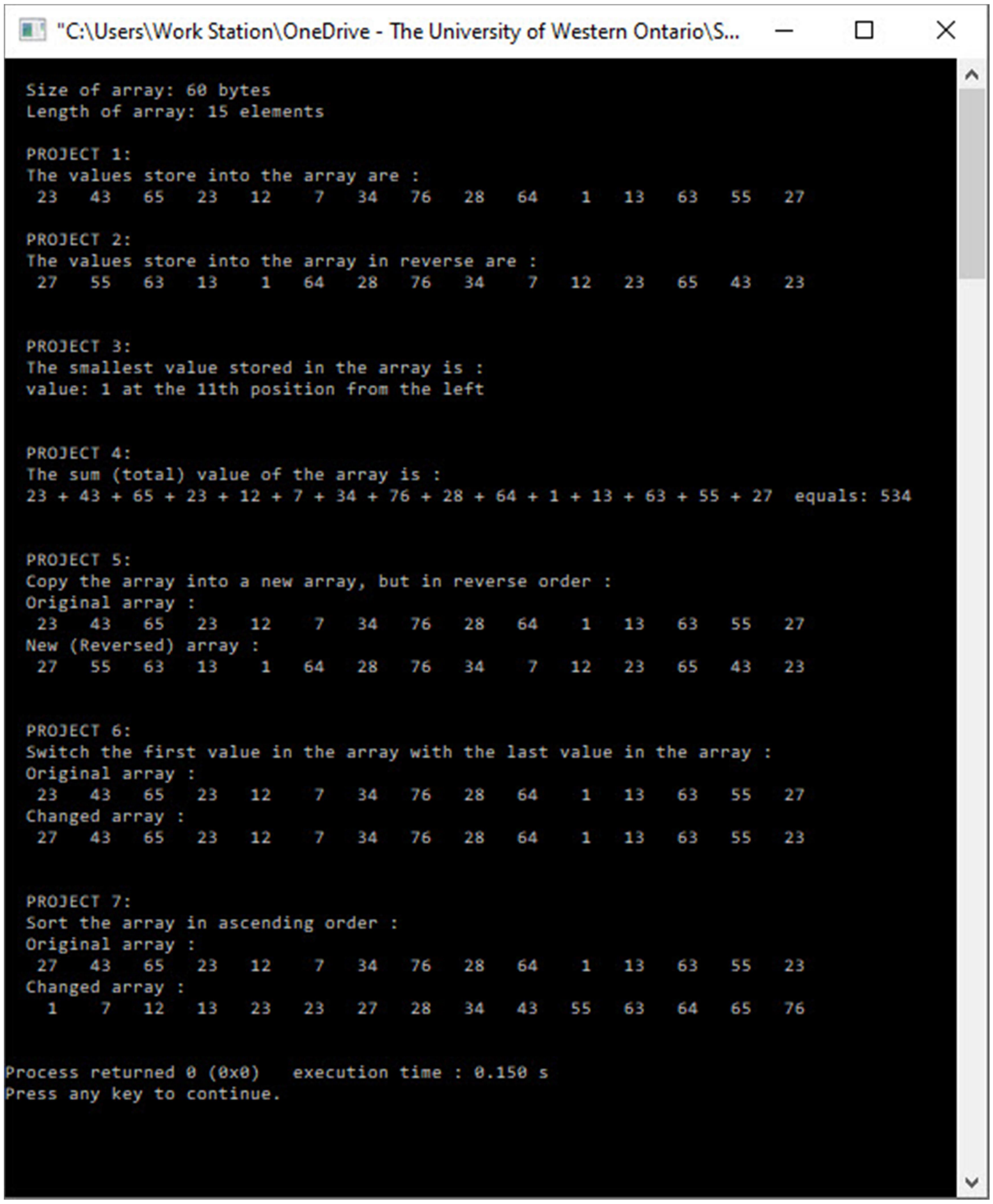
You must write this single program so that it is generic enough where the user can simply add more values to the array definition and it will still run without any other change (zero (0) – NO other change) to the code.

Project 8: Add five new, random values to the array definition
The only line of code to be changed in our program is the array declaration/definition line:

```
int array[] = {23,43,65,23,12,7,34,76,28,64, ?, ?, ?, ?, ?};
```

Replace each question mark above (?) with any unique (not already in the array) value of your choosing. (i.e. pick any five values to complete the array definition – NOT the ones shown here).

Your output should now look like:



"C:\Users\Work Station\OneDrive - The University of Western Ontario\S...

```
  Size of array: 60 bytes
  Length of array: 15 elements

  PROJECT 1:
  The values store into the array are :
   23   43   65   23   12    7   34   76   28   64    1   13   63   55   27

  PROJECT 2:
  The values store into the array in reverse are :
   27   55   63   13    1   64   28   76   34    7   12   23   65   43   23


  PROJECT 3:
  The smallest value stored in the array is :
  value: 1 at the 11th position from the left


  PROJECT 4:
  The sum (total) value of the array is :
  23 + 43 + 65 + 23 + 12 + 7 + 34 + 76 + 28 + 64 + 1 + 13 + 63 + 55 + 27  equals: 534


  PROJECT 5:
  Copy the array into a new array, but in reverse order :
  Original array :
   23   43   65   23   12    7   34   76   28   64    1   13   63   55   27
  New (Reversed) array :
   27   55   63   13    1   64   28   76   34    7   12   23   65   43   23


  PROJECT 6:
  Switch the first value in the array with the last value in the array :
  Original array :
   23   43   65   23   12    7   34   76   28   64    1   13   63   55   27
  Changed array :
   27   43   65   23   12    7   34   76   28   64    1   13   63   55   23


  PROJECT 7:
  Sort the array in ascending order :
  Original array :
   27   43   65   23   12    7   34   76   28   64    1   13   63   55   23
  Changed array :
    1    7   12   13   23   23   27   28   34   43   55   63   64   65   76


Process returned 0 (0x0)   execution time : 0.150 s
Press any key to continue.
```

note: You can NOT use the numbers shown above. The must be number you come up. But, make sure none of the values repeat. They MUST be different values.

Increase the array by adding the five numbers and them compile the program. You will lose all marks on this project if there is any other change, no matter how small, to any other part of the program. The purpose is to provide experience writing generic code.

Submission Instructions:

You must upload and submit, via the CS1037A OWL Web Site, the following Two (2) files:

yourUWOAccountName_Asn02.cbp
main.c

(example: assume my UWO email is kdoit373@uwo.ca
i.e. if my email is – **kdoit373@uwo.ca** then my ID will be – **kdoit373**
So, my UWO Account Name is: **kdoit373** and this assignment is **Asn02**.
therefore, my file names that are to be used for submission are:
kdoit373_Asn02.cbp -and- **main.c**

It is the student's responsibility to ensure the work was submitted and posted in OWL.
OWL replies with a summation verification email (every time).

Any assignment **not** submitted correctly will **not** be graded.

NOTE: ONLY these files are to be submitted.
Do **NOT** submit any of the other files or folders.
Marks will be deducted if you submit anything more than just these two (2) files.
You **MUST** submit a .cbp Project File that can be opened in Code::Blocks
- if you could not get Code::Blocks to work on your computer, use the computers in the lab or in Middlesex College.

The teaching assistant grading your assignment will compile and run your program.
If the program does not compile, the TA will NOT attempt to correct or fix your program so it will run.

AND AGAIN PLEASE: Do not cheat or copy.
Remember, these are NOT community projects but are expected to be completed individually.
Remember the Spanish phrase from the first assignment.