CSCI 1300 CS1: Starting Computing
Ashraf, Cox, Spring 2020
Homework 5
Due: Saturday, February 29, by 6 pm
(5 % bonus on the total score if all three parts are submitted by 11:59 pm February 28)

Objectives

- Understand arrays and vectors.
- Be able to pass arrays to functions

You can find **hw5 note: array and vector** on Moodle.

Submissions

- **Conceptual reviews(mcq)**. There are a few multiple-choice questions to check your conceptual understanding. Don't forget to complete them!
- **C++ files**. All files should be named as specified in each question, and they should compile and run on Cloud 9 to earn full points. TAs will be grading styles of your code and comments. Please see the style guide on Moodle and the summary note on Moodle. At the top of each file, write your name with the following format:

```
// CS1300 Spring 2020
// Author: Punith Sandhu
// Recitation: 123 – Favorite TA
// Homework 5 - Problem # …

/*
 * This function converts a temperature in fahrenheit to celsius
 * and prints the equivalence.
 * Parameters: fahrenheit - degrees fahrenheit
 * Return: equivalent temperature in celsius
*/

double fahrenheit_to_celsius(double fahrenheit){
     double celsius = (fahrenheit - 32) * (5/9.0);
     return celsius;
}
```

  For each question, create a program that calls the function in the main. Here is an example.

- **Code runner**. Your program will be graded by the code runner. You can modify your code and re-submit (press Check again as many times as you need to, up until the assignment due date.

Questions

Question 1(5pt): array pilgrimage

Write a program `arrayPilgrimage.cpp` to declare and populate the four arrays listed below.

- `temps` - an array of 10 floating point numbers (type `double`) initialized with -459.67 (absolute zero in Fahrenheit)
- `colors` - an array of the strings “`Red`”, “`Blue`”, “`Green`”, “`Cyan`”, and “`Magenta`”, in that order.
- `sequence` - an array of the first 100 positive integers in order; `1`, `2`, `3`, `4`, … etc.
- `letters` - an array of all uppercase and lowercase characters in order, `A`, `a`, `B`, `b`, `C`, `c`, … Hint: the ASCII table will be helpful here!

In order to test that you correctly created and populated the arrays, print the values of each of their elements, in order, one element per line. **Hint:** Use a loop to traverse each array.

Sample run

```
-459.67
-459.67
…
-459.67

Red
Blue
…
Magenta

1
2
…
100

A a
B b
…
Z z
```

The file should be named as `arrayPilgrimage.cpp`. Don’t forget to head over to the code runner on Moodle and paste your solution in the answer box!

Question 2(10pt): Array stats
Write a function **stats** that takes an array and the number of elements in the array. Then, it computes and prints the minimum value, maximum value, and the average value of all the values in the array. The output should be formatted with two-digit precision.

Function specifications:
- The function name: **stats**
- The function parameters (in this order):
    - An array, `double`
    - The number of elements stored in the array, `int`
- The function should not return anything.

Sample run 1

```
Test 1: 10.4 3.2 1.4 5.1 6.7
Min: 1.40
Max: 10.40
Avg: 5.36
```

Sample run 2

```
Test 2: 1 2 3
Min: 1.00
Max: 3.00
Avg: 2.00
```

Sample run 3

```
Test 3: -1.2 -12.8 5 10.4 11 2.2 -1
Min: -12.80
Max: 11.00
Avg: 1.94
```

The file should be named as `arrayStats.cpp`. Don't forget to head over to the code runner on Moodle and paste only your function in the answer box! In the main, make sure that you call the function.

Question 3(10pt): Insert into a sorted array
Write a function **insert** that takes an array of integers in ascending order, the number of integers currently in the array, the total size of the array, and an integer to insert into the array. The function should insert the given integer into the array such that the array remains in sorted order. If the array is already full, the array should remain unchanged. The function should return the new number of elements in the array.

Function specifications:

- The function name: **insert**
- The function parameters (in this order):
  - An array, `int`
  - The number of elements stored in the array, `int`
  - The size of the array, `int`
  - The new element to be added, `int`
- The function returns the number of elements after inserting the element

Sample run 1

```
Before Insertion: 1,1,3,4,5,5,5,6
After Insertion:  1,1,2,3,4,5,5,5,6
```

Sample run 2

```
Before Insertion: 1,1,3,4,5,5,5,6
After Insertion:  0,1,1,3,4,5,5,5,6
```

The file should be named as `insert.cpp`. Don't forget to head over to the code runner on Moodle and paste only your function in the answer box! In the main, make sure that you call the function.

Question 4(10pt): sum of the max value found in each row

Write a function **maxSum** that takes two parameters; a 2D integer array with 10 columns, and the number of rows. The function must return the sum of the maximum value from each row.

Function specifications:

- The function name: **maxSum**
- The function parameters (in this order):
  - A 2D **integer** array with 10 columns
  - The number of rows, **`int`**
- The function returns the sum of the maximum value from each row as, an integer

Sample run 1

```
Given matrix:
  9  2  2  3  5  7  9  7  6  8
  6  8  3  9  0  2  3  3 10  6
max sum: 19
```

Explanation: 9 (the highest value in the 1st row) + 10 (the highest value in the 2nd row) = 19.

Sample run 2

```
Given matrix:
  5  2  8 -8  9  9  8 -5 -1 -5
  4  1  8  0 10  7  6  1  8 -5
 11  2 -3  8  7 10  0  3  9 11
max sum: 30
```

Explanation: 9 (the highest value in the 1st row) + 10 (the highest value in the 2nd row) + 11 (the highest value in the 3rd row) = 30.

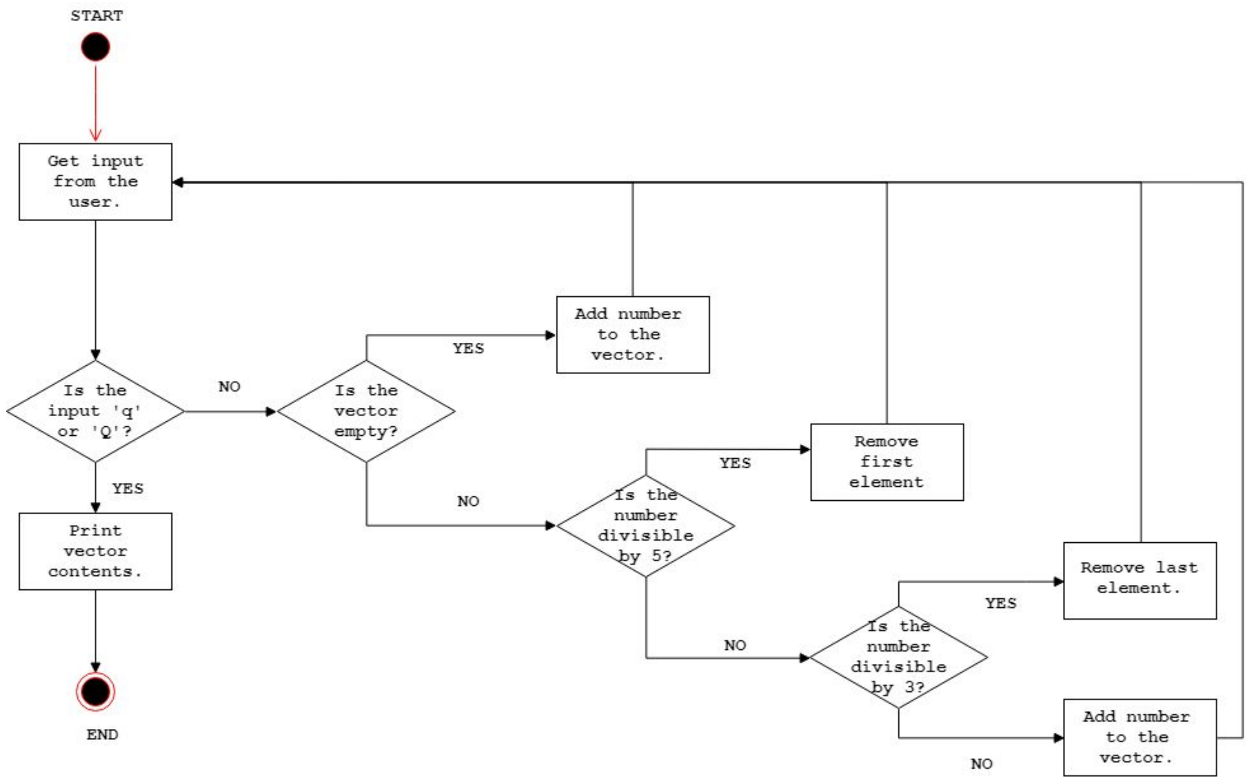
The file should be named as `maxSum.cpp`. Don't forget to head over to the code runner on Moodle and paste only your function in the answer box! In the main, make sure that you call the function.

Question 5(10pt): Vector
Vectors are similar to arrays, but they can grow dynamically in an efficient way. Write a program that asks the user to repeatedly enter positive integers and stop when the user enters -1.

You need to use a vector in this question. At the beginning of the program, the vector is empty. Then, as the user enters values, the vector will be updated based on the following conditions in this order:

- If the vector is empty, insert the user input value into the vector
- If the vector is not empty and the input value is divisible by 5, then remove an element from the front of the vector
- If the vector is not empty and the user input value is divisible by 3, then remove an element from the end of the vector
- Otherwise, insert the input value at the end of the vector.

After the user is done entering values, your program should display all elements in the vector, in order, separated by spaces. On the next line, it also displays the minimum value and the maximum values in the vector.



Sample run 1 (**bold** is user input)

```
Please enter a number:
3
Please enter a number:
4
Please enter a number:
7
Please enter a number:
5
Please enter a number:
8
Please enter a number:
10
Please enter a number:
-1
The elements in the vector are: 7 8
Min = 7
Max = 8
```

The file should be named as `vector.cpp`. Don't forget to head over to the code runner on Moodle and paste your solution in the answer box!

Question 6(15pt): split a string

When you're processing data, it's useful to break up a text string into pieces using a delimiter. Write a function **split** that takes a string, splits it at every occurance of a delimiter, and then populates an array of strings with the split pieces, up to the provided maximum number of pieces.

Function specifications:

- The function name: **split**
- The function parameters (in this order):
    - The string to be split
    - A separator, character, which marks where the string should be split up
    - An array of string, where the split-apart string pieces will be stored
    - The size of the array, `int`
- The function returns the number of pieces the string was split into, as an integer. If the string is split into more pieces than the size of the array, then it returns -1.
- The function should not print anything

Sample run 1 (**bold** is user input; the array size is 4)

```
Enter the text: Apples,Oranges,Bananas
Enter the separator: ,
Return value: 3
arr[0] = Apples
arr[1] = Oranges
arr[2] = Bananas
```

Sample run 2 (**bold** is user input; the array size is 4)

```
Enter the text: 2020//03//11
Enter the separator: /
Return value: 3
arr[0] = 2020
arr[1] = 03
arr[2] = 11
```

Sample run 3 (**bold** is user input; the array size is 4)

```
Enter the text: ,,Tokyo,Bangalore,Boulder,London,Seattle,
Enter the separator: ,
Return value: -1
arr[0] = Tokyo
arr[1] = Bangalore
arr[2] = Boulder
arr[3] = London
```

There are 5 cities, but the size of the array is 4. Since it's more than the size of the array, the split function returns -1.

The file should be named as `split.cpp`. Don't forget to head over to the code runner on Moodle and paste only your function in the answer box! In the main, make sure that you call the function.

Extra credit(10pt): character frequency
Write a program that asks for a string. Then, it populates the counts of each character in the given string. To make it simple, the string only contains lowercase alphabetical characters from a to z. After getting the counts, print the characters and their counts in the alphabetical order, if they occur at least once in the string.

Expected output 1 (**bold** is user input)

```
Enter a word:
colorado
a: 1
c: 1
d: 1
l: 1
o: 3
r: 1
```

The file should be named as `charCounts.cpp`. Don't forget to head over to the code runner on Moodle and paste your solution in the answer box!

Homework 5 checklist

Here is a checklist for submitting the assignment:
1. Complete the **conceptual reviews(mcq)**
2. Complete the code **Homework 5 CodeRunner**
3. Submit one zip file to **Homework 5**. The zip file should be named, **hmwk5_lastname.zip**. It should have the following 6 files:
   - **arrayPilgrimage.cpp**
   - **arrayStats.cpp**
   - **insert.cpp**
   - **maxSum.cpp**
   - **vector.cpp**
   - **split.cpp**

Homework 5 points summary

Criteria	**Pts**
Conceptual reviews (MCQ)	10
CodeRunner (problem 1 - 6)	60
C++ file submission (compiles and runs, style and comments)	30
Recitation attendance (Feb 24, Feb 25)*	-30
Total	100
5% early submission bonus	+5%
Extra credit: character frequency	+10 pt

* if your attendance is not recorded, you will lose points. Make sure your attendance is recorded on Moodle.