CSCI 1300 - Starting Computing
Instructor: Fleming
Recitation 10

This assignment is due **Friday, November 16th, by 11:55 pm**

- ***All components*** **(Cloud9 workspace and moodle quiz attempts) must be completed and submitted by Friday, November 16th, by 11:55 pm for your solution to receive points.**
- **Recitation attendance is required to receive credit.**

Please follow the same submission guidelines outlined in Homework 3 description regarding Style, Comments and Test Cases. Here's a review below on what you need to submit for Recitation 10.

Develop in Cloud9: For this recitation assignment, write and test your solution using Cloud9.

Submission: All three steps must be fully completed by the submission deadline for your homework to receive points. Partial submissions will not be graded.

1. ***Make sure your Cloud 9 workspace is shared with your TA:*** Your recitation TA will review your code by going to your Cloud9 workspace. *TAs will check the last version that was saved before the submission deadline*.
    - Create a directory called **Rec10** and place all your file(s) for this assignment in this directory.
    - Make sure to *save* the final version of your code (File > Save). Verify that this version displays correctly by going to File > File Version History.
    - The file(s) should have all of your functions, test cases for the functions in main function(s), and adhere to the style guide. Please read the **Test Cases** and **Style and Comments** sections included in the **Homework 3** write up for more details.
2. ***Submit to the Moodle Autograder:*** Head over to Moodle to the link **Rec 10**. You will find one programming quiz question for each problem in the assignment. Submit your solution for the first problem and press the Check button. You will see a report on how your solution passed the tests, and the resulting score for the first problem. You can modify your code and re-submit (press *Check* again) as many times as you need to, up until the assignment due date. Continue with the rest of the problems.

CSCI 1300 - Starting Computing
Instructor: Fleming
Recitation 10

Introduction to Vectors

Let's start with something we already know about – arrays.

To recap, an array is a contiguous series that holds a *fixed number of values* of the same datatype.

A vector is a template class that uses all of the syntax that we used with vanilla arrays, but adds in functionality that relieves us of the burden of keeping track of memory allocation for the arrays. It also introduces a bunch of other features that makes handling arrays much simpler.

How Vectors Work- Syntax and Usage

First things first. We need to include the appropriate header files to use the vector class.

```
#include <vector>
```

We can now move on to declaring a vector. This is general format of any vector declaration:

```
vector <datatype_here> name(size);
```

The size field is optional. Vectors are *dynamically-sized*, so the size that you give them during initialization isn't permanent- they can be resized as necessary.

Let's look at a few examples

```
//Initializes a vector to store int values of size 10.
vector <int> vec1(10);

//Initializes an empty vector that can store strings.
vector <string> vec2;
```

You can access elements of a vector in the same way you would access elements in an array, for example `array[4]`. Remember, indices begin from 0.

CSCI 1300 - Starting Computing
Instructor: Fleming
Recitation 10

You can find a quick reference to the functions available in C++ vector class in a nice PDF form here, but following are the ones you will need in this recitation:

All these functions belong to the vector *class*, so you need to call them on an instance of a vector. (vec1.size(), for example).

- **`size()`** returns the size of a vector.
- **`at()`** takes an integer parameter for index and returns the value at that position

Adding elements to the vector is done primarily using two functions

- **`push_back()`** takes in one parameter (the element to be added) and appends it to the end of the vector.

```
vector <int> vector1; // initializes an empty vector
vector1.push_back(1); //Adds 1 to the end of the vector.
vector1.push_back(3); //Adds 3 to the end of the vector.
vector1.push_back(4); //Adds 4 to the end of the vector.
cout<< vector1.size(); //This will print the size of the
vector - in this case, 3.
//vector1 looks like this: [1, 3, 4]
```

- **`insert()`** can add an element at some position in the middle of the vector.

```
//vectorName.insert(vectorName.begin() + position, element)
vector1.insert(vector1.begin() + 1, 2);
cout << vector1.at(1) << endl; // 2 is at index=1
//vector1 looks like this: [1, 2, 3, 4]
```

Here, the **begin** function returns an iterator to the first element of the vector. You can think of it as an arrow pointer that points to the memory location just before the first element. The **`begin()`** would thus refer to the first element and **`begin()+k`** would refer to the kth element in the vector, k starting at 0.

Elements can also be removed.

- **`pop_back()`** deletes the last element in the vector

- **`erase()`** can delete a single element at some position

```
//vector_name.erase(vector_name.begin() + position)
vector1.erase(vector1.begin() + 0);
cout << vector1.at(0) << endl; //2 is at index=0
//vector1 looks like this: [2, 3, 4]
```

The vector class has many more functions so look at: http://www.broculos.net/2007/11/c-stl-vector-class-cheatsheet.html if you are interested in learning more. (You may want to use vectors instead of arrays for project 3).

CSCI 1300 - Starting Computing
Instructor: Fleming
Recitation 10

Problem 1

Write a function that takes in an integer vector as a parameter. The function should prompt the user to enter integer values and fill the vector with those values at the rear of the vector. It should continue to prompt the user until the user enters a value less than or equal to 0.

You should use the following function signature, which has been pre-loaded for you:

```
void modifyVector(vector<int>& v)
```

You should use the following prompt for user input:

```
Please enter an integer value:
```

Rules for filling the vector:

- If the number is less than or equal to 0, end.
- Otherwise, if the vector is empty, insert the user input value into the vector.
- Otherwise, if the user input value is divisible by 5, then remove an element from the front(beginning) of the vector.
- Otherwise, if the user input value is divisible by 3, then remove an element from the end of the vector.
- Otherwise, insert the user input value at the end of the vector.

Here are those rules in flowchart form:

CSCI 1300 - Starting Computing
Instructor: Fleming
Recitation 10