SFU SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

ENSC 251 D100 – Software Design and Analysis for Engineers (4 sem. hrs.)
Fall 2017

HW 1

Assigned	September 20, 2017
Due	October 4, 2017 @ 9:00am.

This is a team assignment.

- You are free to choose your team partner. A team should consist of no more than 2 people. If you cannot find a partner and there isn't one available, please feel free to work on your own. You can also choose to work on your own in a team.
- You may consult with professor and TA about any aspect of the assignment.
- You may consult with other students only in a general way, e.g., about debugging or C++ issues, or questions about wording on the assignment.
- You cannot actively work with another student that is not in your team in this assignment.

1. Specifications

Question 1 (Savitch, 10th, Ch. 10, page 617, Question 10)

The U.S. Postal Service printed a bar code on every envelope that represented a five- (or more) digit zip code using a format called POSTNET (this format was deprecated in favor of a new system, OneCode, in 2009). The bar code consists of long and short bars as shown:

For this program, we will represent the bar code as a string of digits. The digit 1 represents a long bar, and the digit 0 represents a short bar. Therefore, the bar code would be represented in our program as

```
11010010100010101100001O011
```

The first and last digits of the bar code are always 1. Removing these leaves 25 digits. If these 25 digits are split into groups of 5 digits each, we have

```
10100 10100 01010 11000 01001
```

Next, consider each group of 5 digits. There will always be exactly two 1s in each group of digits. Each digit stands for a number. From left to right, the digits encode the values 7, 4, 2, 1, and 0. Multiply the corresponding value with the digit and compute the sum to get the final encoded digit for the zip code. The table below shows the encoding for 10100.

Bar code digits	1	0	1	0	0
Value	7	4	2	1	0
Product of digit * Value	7	0	2	0	0

Zip Code Digit = 7 + 0 + 2 + 0 + 0 = 9

Repeat this for each group of 5 digits and concatenate to get the complete zip code. There is one special value. If the sum of a group of 5 digits is 11, then this represents the digit 0 (this is necessary because with two digits per group it is not possible to represent zero). The zip code for the sample bar code decodes to 99504. Although the POSTNET scheme may seem unnecessarily complex, its design allows machines to detect if errors have been made in scanning the zip code.

Write a zip code class that encodes and decodes 5-digit bar codes used by the U.S. Postal Service on envelopes. The class should have two constructors. The first constructor should input the zip code as an integer, and the second constructor should input the zip code as a bar code string consisting of 0s and 1s, as described above. Although you have two ways to input the zip code, internally, the class should store the zip code using only one format (you may choose to store it as a bar code string or as a zip code number). The class should also have at least two public member functions, one to return the zip code as an integer, and the other to return the zip code in bar code format as a string. All helper functions should be declared private.

Embed your class definition in a suitable test program (See section 5.3 – 5.5). Your program should print an error message if an invalid bar code is passed to the constructor.

Question 2 (Savitch, 10th, Ch. 11, page 696, Question 1)
In Chapter 8 we discussed vectors, which are like arrays that can grow in size. Suppose that vectors were not defined in C++. Define a class called `VectorDouble` that is like a class for a vector with base type `double`. Your class `VectorDouble` will have a private member variable for a dynamic array of `doubles`. It will also have two member variables of type `int`; one called `max_count` for the size of the dynamic array of `doubles`; and one called count for the number of array positions currently holding values. (`max_count` is the same as the capacity of a vector; count is the same as the size of a vector.)

If you attempt to add an element (a value of type `double`) to the vector object of the class `VectorDouble` and there is no more room, then a new dynamic array with twice the capacity of the old dynamic array is created and the values of the old dynamic array are copied to the new dynamic array.

Your class should have all of the following:

- Three constructors: a default constructor that creates a dynamic array for 50 elements, a constructor with one int argument for the number of elements in the initial dynamic array, and a copy constructor.
- A destructor.
- A suitable overloading of the assignment operator =.

- A suitable overloading of the equality operator ==. To be equal, the values of count and the count array elements must be equal, but the values of max_count need not be equal.
- Member functions push_back, capacity, size, reserve, and resize that behave the same as the member functions of the same names for vectors.
- Two member functions to give your class the same utility as the square brackets: value_at(i), which returns the value of the ith element in the dynamic array; and change_value_at(d, i), which changes the double value at the ith element of the dynamic array to d. Enforce suitable restrictions on the arguments to value_at and change_value_at. (Your class will not work with the square brackets. It can be made to work with square brackets, but we have not covered the material which tells you how to do that.)

Also, you need to include a test driver program to show case the functionalities of the class. (See section 5.3 – 5.5)

2. Submission

You can use the example zip file from lab 1 as a starting point. Create *.cpp file as needed. Modify the makefile such that it will compile your code into a binaries executable.

a) Create a directory with your name, e.g. "\LastnameFirstname", where Lastname is student's last name and Firstname is the first name.
b) Then create subdirectories with various questions, e.g. "\q1", "\q2". Each subdirectory should be self contain and has it's own makefile etc.
c) Save the files (*.cpp, other files, and makefile) in these directories. Uses these files as a starting point to write the following program.

Then Zip up the directory "\LastnameFirstname" and the files within this director into a zip file "2017-3-ENSC251-LastnameFirstname.zip." Submit the zip file to Canvas before the deadline. Since this is a group project, use the name of one of the team member.

Appendix A: Rubric for marking

Criteria	Ratings				Pts
Program Specifications / Correctness	**Excellent** - No errors, program always works correctly and meets the specification(s). 50.0 pts	**Adequate** - Minor details of the program specification are violated, program functions incorrectly for some inputs. 40.0 pts	**Poor** - Significant details of the specification are violated, program often exhibits incorrect behavior. 30.0 pts	**Not met** - Program only functions correctly in very limited cases or not at all. 0.0 pts	50.0 pts
Readability	**Excellent** - No errors, code is clean, understandable, and well-organized. 20.0 pts	**Adequate** - Minor issues with consistent indentation, use of whitespace, variable naming, or general organization. 16.0 pts	**Poor** - At least one major issue with indentation, whitespace, variable names, or organization. 12.0 pts	**Not met** - Major problems with at three or four of the readability subcategories. 0.0 pts	20.0 pts
Documentation	**Excellent** - No errors, code is well-commented. 20.0 pts	**Adequate** - One or two places that could benefit from comments are missing them or the code is overly commented. 16.0 pts	**Poor** - File header missing, complicated lines or sections of code uncommented or lacking meaningful comments. 12.0 pts	**Not met** - No file header or comments present. 0.0 pts	20.0 pts
Code Efficiency	**Excellent** - No errors, code uses the best approach in every case. 5.0 pts	**Poor** - Code uses poorly-chosen approaches in at least one place. 3.0 pts	**Not met** - Many things in the code could have been accomplished in an easier, faster, or otherwise better fashion 0.0 pts		5.0 pts
Assignment Specifications	No errors 5.0 pts	Minor details of the assignment specification are violated, such as files named incorrectly or extra instructions slightly misunderstood 3.0 pts	Significant details of the specification are violated, such as extra instructions ignored or entirely misunderstood 0.0 pts		5.0 pts
Total Points: 100.0					

-END-