**CYBR210 – Problem Set 8**

Name: Steven Heffernan

1. (5 Points) Define a class template. When would a programmer want to use class templates? (Page 831)

**A class template allows you to write a class definition with “blanks” left in the definition to be filled in by the calling code. You want to use class templates when instantiating multiple classes of data types.**

1. (5 Points) Define a function template. (Page 833)

**A C++ language construct that allows the compiler to generate multiple versions of a function by allowing parameterized data types.**

1. (10 Points) Define function overloading. Provide a simple example with two function definitions where the functions are overloaded. (Page 842)

**Function overloading is the use of the same name for different functions, distinguished from each other by their parameter lists.**

**void Print(int number)**

**{**

**cout << "\*\*\*Debug" << endl;**

**cout << "Value is " << number << endl;**

**}**

**void Print(float realValue)**

**{**

**cout << "\*\*\*Debug" << endl;**

**cout << "Value is " << realValue << endl;**

**}**

1. (15 Points) In a well-written paragraph, discuss the use of exceptions and exception handling in programming. Your discussion should include definitions for exception, exception handler, throw signal, and catch process. Discuss why exceptions are helpful for handling runtime errors, such as division by zero.

**Exception handling is a way of identifying possible events that you might not be able to think of addressing until an exception occurs. YOu can also use exception handling to deal with known exceptions, like input validation. You can use exception handlers to address any possible exceptions and deal with them accordingly. Once an exception is thrown or raised, you can use catch processes to manage how you want them handled or managed.**

**Exceptions are helpful for handling runtime errors like division by zero. For example, if you're processing fractions and you want to ensure your program follows the mathematical principals of division, you will want to check that the fraction is divisible by zero. You can write a boolean function to tell if the fraction is divisible by zero, thus meeting the requirements. Then, if the fraction doesn't meet the criteria, you can throw an error, maybe a string that informs the user that the fraction input is not correct.**

**HINT**: For problems 5 -7, look at the quick check problems from page 858 and their solutions on page 880 as a template.

1. (5 Points) Write a declaration for a user-defined exception type named **BadYear**.

**class BadYear**

**{};**

1. (10 Points) Write a void function **GetYear** that prompts for and inputs the year the operator was born (type **int**) from standard input. The function returns the user’s birth year through the parameter list (use pass by reference) unless the user enters an invalid year, in which case a **BadYear** exception is thrown. To test for a bad year, think about the range of acceptable years. It must be 4 digits (i.e. 1982) and it cannot be greater than the current year (the user obviously cannot be born in a calander year that has yet to happen!).
2. (10 Points) Write a try-catch statement that calls the **GetYear** function defined above. If a **BadYear** exception is thrown, print an error message and rethrow the exception to a caller; otherwise, execution should just continue as normal.
3. (15 Points) Consider our **Fraction** class that we wrote for problem set 7. Implement an exception handling design in the parameterized constructor to catch zeros in the denominator. Show the new exception class you will need to put in the header (specification) file. Show the definition for your parameterized constructor that throws the exception if the denominator is zero. Finally, show how to use the try-catch statement in the driver program when instatiating a new paramerized Fraction object. You do not need to show entire programs, only the requested lines of source code. (See the Date ADT example on pages 859-863)
4. The **vector** Template (Pages 898 – 903)
   1. (3 Points) What is the underlying implementation provided by the **vector** class template?
   2. (3 Points) Show an example call to a vector constructor that creates a vector holding 15 integers.

**vector<int> intVec(15);**

* 1. (3 Points) Show an example call to a vector constructor that creates a vector holding zero strings.

**vector<string> noSizeVec;**

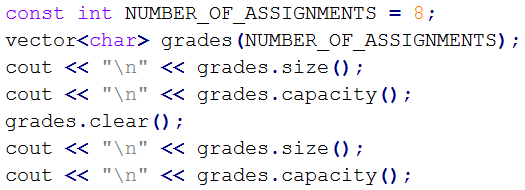
* 1. (4 Points) Why would we want to declare a vector with no space in it?

**When you’re unsure of the volume of data that it will be ingesting.**

* 1. (4 Points) If we know the maximum size of a vector in advance, why should we indicate that value in the constructor instead of leaving the vector empty?

**You should always set the value if you know the maximum because vectors can consume a lot of memory and CPU.**

* 1. (4 Points) What does the following code print to standard output?



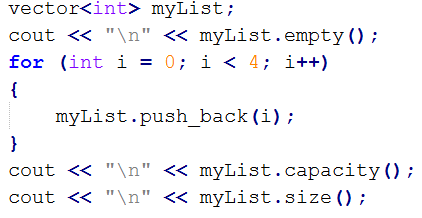
**8**

**8**

**0**

**8**

* 1. (4 Points) What does the following code print to standard output?



**1**

**4**

**4**