**Instructions:**

1. Create a MS VisualStudio solution named **Assignment16**.
2. Create projects according to the assigned problems.
3. Use techniques and experience that you have obtained from previous chapters and this chapter.
4. Code and test your programs. They **MUST** be completed without syntax, logic and run-time errors.
5. Include comments of your name, date, and brief descriptions in all source codes.
6. Compress **Assignment16**folder into **ONE** zipped file.
7. Submit or re-submit your zipped file before its due date&time.

int main()  
{  
   do  
   {  
       switch (menuOption())  
       {  
       case 0: exit(1); break;  
      case 3: Challenge3(); break; //done  
      case 4: Challenge4(); break; //done  
      case 6: Challenge6(); break; //done  
      case 7: Challenge7(); break;  
      case 8: Challenge8(); break;  
      case 12: Challenge12(); break; //10 pts extra  
       default: cout << "\t\tERROR - Invalid option. Please re-enter."; break;  
       }  
       cout << "\n";  
       system("pause");  
  
    } while (true);  
  
   return EXIT\_SUCCESS;  
}

3. Minimum/Maximum Templates

Write templates for the two functions minimum and maximum. The minimum function should accept two arguments and return the value of the argument that is the lesser of the two. The maximum function should accept two arguments and return the value of the argument that is the greater of the two. Design a simple driver program that demonstrates the templates with various data types.

4. Absolute Value Template

Write a function template that accepts an argument and returns its absolute value. The absolute value of a number is its value with no sign. For example, the absolute value of −5 is 5, and the absolute value of 2 is 2. Test the template in a simple driver program.

6. IntArray Class Exception

Chapter 14 presented an IntArray class that dynamically creates an array of integers and performs bounds checking on the array. If an invalid subscript is used with the class, it displays an error message and aborts the program. Modify the class so it throws an exception instead.

7. TestScores Class

Write a class named TestScores. The class constructor should accept an array of test scores as its argument. The class should have a member function that returns the average of the test scores. If any test score in the array is negative or greater than 100, the class should throw an exception. Demonstrate the class in a program.

8. SimpleVector Modification

Modify the SimpleVector class template presented in this chapter to include the member functions push\_back and pop\_back. The push\_back function should accept an argument and insert its value at the end of the array. The pop\_back function should accept no argument and remove the last element from the array. Test the class with a driver program.

12. Specialized Templates

In this chapter, the section Specialized Templates within Section 16.4 describes how to design templates that are specialized for one particular data type. The section introduces a method for specializing a version of the SimpleVector class template so it will work with strings. Complete the specialization for both the SimpleVector and SearchableVector templates. Demonstrate them with a simple driver program.