**Instructions:**

1. Create a MS VisualStudio solution named **Assignment20**.
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 **Assignment20**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 1: Challenge1(); break;  
        case 3: Challenge3(); break;  
        case 4: Challenge4(); break;  
        case 6: Challenge6(); break;  
        case 7: Challenge7(); break;  
        case 8: Challenge8(); break;  
        case 9: Challenge9(); break;  
        case 10: Challenge10(); break;  
        case 11: Challenge11(); break;  
        case 12: Challenge12(); break;  
        default: cout << "\t\tERROR - Invalid option. Please re-enter."; break;  
        }  
        cout << "\n";  
        system("pause");  
    } while (true);

* 1. Iterative Factorial

Write an iterative version (using a loop instead of recursion) of the factorial function shown in this chapter. Test it with a driver program.

1. QuickSort Template

Create a template version of the QuickSort algorithm that will work with any data type. Demonstrate the template with a driver function.

1. Recursive Array Sum

Write a function that accepts an array of integers and a number indicating the number of elements as arguments. The function should recursively calculate the sum of all the numbers in the array. Demonstrate the function in a driver program.  
  
6. Recursive Power Function

Write a function that uses recursion to raise a number to a power. The function should accept two arguments: the number to be raised and the exponent. Assume that the exponent is a nonnegative integer. Demonstrate the function in a program.

1. Sum of Numbers

Write a function that accepts an integer argument and returns the sum of all the integers from 1 up to the number passed as an argument. For example, if 50 is passed as an argument, the function will return the sum of 1, 2, 3, 4, … 50. Use recursion to calculate the sum. Demonstrate the function in a program.

1. isMember Function

Write a recursive Boolean function named isMember. The function should accept two arguments: an array and a value. The function should return true if the value is found in the array, or false if the value is not found in the array. Demonstrate the function in a driver program.

1. String Reverser

Write a recursive function that accepts a string object as its argument and prints the string in reverse order. Demonstrate the function in a driver program.

1. maxNode Function

Add a member function named maxNode to the NumberList class discussed in this chapter. The function should return the largest value stored in the list. Use recursion in the function to traverse the list. Demonstrate the function in a driver program.

1. Palindrome Detector

A palindrome is any word, phrase, or sentence that reads the same forward and backward. Here are some well-known palindromes:

Able was I, ere I saw Elba

A man, a plan, a canal, Panama

Desserts, I stressed

Kayak

Write a bool function that uses recursion to determine if a string argument is a palindrome. The function should return true if the argument reads the same forward and backward. Demonstrate the function in a program.

1. Ackermann’s Function

Ackermann’s Function is a recursive mathematical algorithm that can be used to test how well a computer performs recursion. Write a function A(m, n) that solves Ackermann’s Function. Use the following logic in your function:

If m = 0 then return n + 1

If n = 0 then return A(m−1, 1)

Otherwise, return A(m−1, A(m, n−1))

Test your function in a driver program that displays the following values:

A(0, 0) A(0, 1) A(1, 1) A(1, 2) A(1, 3) A(2, 2) A(3, 2)