```cpp
//  Roster Numbers: 4
//
//  Authors: Cody Blakeney
//  Due Date: 4/4/2016
//  Programming Assignment Number 6
//
//  Spring 2016 - CS 3358 - 253
//
//  Instructor: Husain Gholoom
//
//  The purpose of this program is to use recursions to write functions
//  that manipulate digits in integers

#include <iostream>

                std;

/*
    Increasing Order returns true if digits are in
    increasing order from left to right.
*/
bool increasingOrder(int);
/*
    reverse order returns an integer in the reverse of its original digit
    sequence.
*/
int reverseOrder(int, int reverse = 0);
/*
    sumDigits returns the sum of the digits in the integer sequence.
    its argument n is the integer to sum.
*/
int sumDigits(int);
/*
    sumSquares finds the sum of the square of all numbers from 1 to n
*/
int sumSquares(int);
/*
    disVert displays an integer by its digits vertically
*/
void disVert(int);

int main(){

    int choice;  // user var that chooses from the menu
    int input;  // integer to be evaluated


    cout << "\n*** Welcome to My Program Using Recursions  *** \n\n" ;
    cout << "The function of this program is to \n";
    cout << "accepts from the keyboard\n" ;
    cout << "a positive integer that is  > 9.\n";
    cout << "The program then does the following  :\n\n";
    cout << "    1- Returns true if the digits of that integer n\n";
    cout << "       are in increasing order; otherwise ," << endl;
    cout << "       the function returns false." << endl;
    cout << "    2- Returns the numbers with the digits reversed.\n";
    cout << "    3- Returns  the sum of the digits of the integer.\n";
    cout << "    4- Returns the sum of squares of the numbers from " <<
            "0 to the number n." << endl;
    cout << "    5- Displays the number vertically .\n\n";


      {  // program running loop

        cout << "\nSelect from the following menu\n";
        cout << "1.  Enter a positive integer >  9.\n";
        cout << "9.  Terminate the program.\n";
```

```
        cin >> choice;
             (cin.fail()){ //makes sure an integer was the input
        cin.clear();       //and fixes cin if other type
        cout << "You did not enter an integer. " << endl ;
        cin.ignore(256, '\n');
        }
          (choice != 1 && choice != 9)   // if anything other than 1, 9 is
            cout << "Invalid Option.\n"; // picked, error is displayed


          (choice == 1){

                cout << "Enter a positive integer >  9.    ";
                cin >> input;
                     (cin.fail()){ //makes sure an integer was the input
                        cin.clear(); // and fixes cin if other type
                        cout << "You did not enter an integer. " << endl ;
                        cin.ignore(256, '\n');
                }

                  (input <= 9){ // validates user entry
                    cout << "Invalid Number - Number must be > 9\n\n";
                }

                    {

                    cout << "\n\nThe digits of " << input;

                      (increasingOrder(input))
                        cout << " are in increasing order. \n\n";

                        cout << " are not in increasing order.\n\n";

                    cout << "The Original Digits are  " << input << " ---";
                    cout << "  Digits reversed = " << reverseOrder(input);

                    cout << "\n\nSum of digits of the number  " << input <<
                            "  is   =   " << sumDigits(input);
                    cout << "\n\nSum of squares from 0  to  "  << input <<
                            "  =  " << sumSquares(input) << endl << endl;

                    cout << input << " Displayed vertically\n";
                    disVert(input);
                }

        }
    }     (choice != 9); // choice of 9 ends program

    cout << "\n*** program is terminated ***\n";
    cout << "Thank you for using my program using recursions\n";
    cout << "Written by Cody Blakeney\n";

           0;
}

/*
   Increasing Order returns true if digits are in increasing
   order from left to right.
*/
bool increasingOrder(int n){

    // starts by looking at the right most digit
    int lastDigit = n % 10;
    n= n/10;
```

```
    // when zero the integer seq is complete and in increasing order
      (n== 0){
                    ;
    }

    // if the next digit is smaller than the last digit continue to run
    // otherwise it will return false
            (n % 10 < lastDigit){

          (increasingOrder(n)){
                        ;
        }
    }

                     ;
}

/*
    reverse order returns an integer in the reverse of its original digit
    sequence. its argument n is the integer to reverse and reverse is
    the currently reversed section to be passed into the function
    recursively.
*/
int reverseOrder(int n, int reverse){

      (n <= 0)             // once n <= 0 the process is complete
                reverse;

    reverse = reverse * 10 + n % 10; // multiplying by ten shifts the last
                                     // last digit over one to all space for
                                     // a new digit to be added
            reverseOrder(n/10, reverse);
}
/*
    sumDigits returns the sum of the digits in the integer sequence.
    its argument n is the integer to sum.
*/
int sumDigits(int n){

    int sum = 0;
      (n > 0)
        // each recursive call finds the right most digit.
        sum = n % 10 + sumDigits(n / 10);
           sum;
}

/*
    sumSquares finds the sum of the square of all numbers from 1 to n
*/
int sumSquares(int n){

    // each recursive call finds the square of n currently until = 0
    int sum = 0;
      (n > 0)
        sum = (n * n) + sumSquares(n - 1);

           sum;
}

/*
    disVert displays an integer by its digits vertically
*/
void disVert(int n){

      (n > 0){
        disVert(n / 10);
        cout << n % 10 << endl; // this prints the right most digit
```

```
                                // at the current level of the recursion
                                // when the deepest level of recursion is
                                // reached this will be in the correct order
    }
}
```