Lab Quiz 1 Quiz Answer CS111 Summer 2020

The only outside functions you can use are:
a) cout, cin, rand(), srand(), and time()
b) functions you write yourself
c) functions that appear anywhere on this test
d) Each question is worth 10 points each.

```
double SquareRoot(double d)
{
   double squareRoot = 0;

   while (squareRoot * squareRoot < d)
      squareRoot += .01;

   if (squareRoot * squareRoot > d)
      squareRoot -= .01;

   return(squareRoot);
}

#define MAX_VALUE 5
#define START_VALUE 5 // START_VALUE always <= MAX_VALUE
int main()
{
```

Question 1)

```
   // convert do while loop below to while loop
   int i = START_VALUE;
   do
   {
      cout << i << endl;
      i--;
   } while (i >= MAX_VALUE);

   // while loop
   {
      int i = START_VALUE;
      bool firstTime = true;
      while (i >= MAX_VALUE|| firstTime)
      {
         firstTime = false;

         cout << i << endl;
         i--;
      }
```

```
}
```

Question 2)

```
// starting with the value MAX_VALUE repeatedly print the value of the
// variable myVal decreasing it by 0.5 each time,as long
// as myVal remains positive.
double myVal = MAX_VALUE;
while (myVal >= 0)
{
   cout << myVal << endl;
   myVal -= .5;
}
// using as few lines as possible and the SquareRoot function above
// Print the square roots of the first 25 odd positive integers whose
// square root is a whole number (i.e. 3.0 not 3.1)}
// using as few lines as possible and the SquareRoot function above
// Print the square roots of the first 25 odd positive integers whose
// square root is a whole number (i.e. 3.0 not 3.1)}
{
   int i = 0, nNumbers = 0;
   while (nNumbers < 25)
   {
      if (i % 2 == 0)
      {
         i++;
         continue;
      }
      {
         double d = SquareRoot(i);
         int j = d;
         if (j == d)
         {
            cout << "Prime: " << i << " " << j << endl;
            nNumbers++;
         }
      }
      i++;
   }
}
}
```

Question 3)

```
// display time as HH:MM:SS
// DisplayTime(1, 25, 4) ==> 01:25:04
// DisplayTime(11, 5, 44) ==> 11:05:44
void DisplayTime(int hours, int minutes, int seconds)
```

```
{
   if (hours < 10)
      cout << "0";
   cout << hours << ":";
   if (minutes < 10)
      cout << "0";
   cout << minutes << ":";
   if (seconds < 10)
      cout << "0";
   cout << seconds<<endl;
}

// using DisplayTime display new time when adding seconds
//AddSeconds(7, 7, 7, 60); ==> 07:08:07
//AddSeconds(7, 7, 7, 3600); ==> 08:07:07
void AddSeconds(int hours, int minutes, int seconds, int secondsToAdd)
{
   int allSeconds = (hours * 3600) + (60 * minutes) + seconds + secondsToAdd;
   hours = allSeconds / 3600;
   allSeconds -= (hours * 3600);
   minutes = allSeconds / 60;
   allSeconds -= (minutes * 60);
   seconds = allSeconds;
   DisplayTime(hours, minutes, seconds);
}


#define MAX_VALUE 5
#define START_VALUE 5 // START_VALUE always <= MAX_VALUE
int main()
{
```

Question 1)

```
   // convert do while loop below to a for loop
   int j = MAX_VALUE;
   do
   {
      cout << j << endl;
      j--;
   } while (j >= 0);

   for(int j= MAX_VALUE;j>=0;j--)
      cout << j << endl;
}
```

Question 2)

// using a double for loop display the figure below where the

```
// number of rows and columns == size
//DisplayFigure(10)==>
/*
**********
 *
  *
   *
    *
     *
      *
       *
        *
         *
*/
void DisplayFigure(int size)
{
   for (int row = 1; row <= size; row++)
   {
      for (int col = 1; col <= size; col++)
      {
         if (row == 1 || row == col)
            cout << "*";
         else
            cout << " ";
      }
      cout << endl;
   }
}
```

Question 3)

```
// in as few lines as possible convert to roman numerals the num
// passed in. The input is limited to 1..39
// output the value passed in as a Roman Numeral
/// ConvertToRoman(1) ==> I
//ConvertToRoman(1) ==> II
//ConvertToRoman(4) ==> IV
//ConvertToRoman(5) ==> V
//ConvertToRoman(9) ==> IX
//ConvertToRoman(30) ==> XXX
void ConvertToRoman(int n)
{
   while (n >= 10)
   {
      cout << "X";
      n -= 10;
   }
```

```
    while (n >= 9)
    {
        cout << "IX";
        n -= 9;
    }
    while (n >= 5)
    {
        cout << "V";
        n -= 5;
    }
    while (n >= 4)
    {
        cout << "IV";
        n -= 4;
    }
    while (n >= 1)
    {
        cout << "I";
        n -= 1;
    }
}
int main()
{
```

Question 1)

```
    // using a do while loop output all the upper case letters 'A' - 'Z'
    char c = 'A';
    do
    {
        cout << c;
        c++;
    } while (c <= 'Z');

    // in the SquareRoot function above why is the line below necessary?
    //if (squareRoot * squareRoot > d)
    //    squareRoot -= .01;
    Because the while loop may go one unit above the actual square
}
```

Question 2)

```
// in the least amount of lines return the number
// of days in a month
```

Month Number	Month	In 3 letters	Days in Month

1	January	Jan	31
2	February	Feb	28 (29 in leap years)
3	March	Mar	31
4	April	Apr	30
5	May	May	31
6	June	Jun	30
7	July	Jul	31
8	August	Aug	31
9	September	Sep	30
10	October	Oct	31
11	November	Nov	30
12	December	Dec	31

```
// in the least amount of lines return the number
// of days in a month
// every year divisible by 400 or 4 (but not 100) has 29 days for February
// every other years has 28 days for February
// NumberOfDaysInMonth(12, 2020) ==> 31
// NumberOfDaysInMonth(2, 2020) ==> 29
// NumberOfDaysInMonth(2, 2021) ==> 28
int  NumberOfDaysInMonth(int month, int year)
{
   //leap year condition, if month is 2
   if (month == 2)
   {
      if ((year % 400 == 0) || (year % 4 == 0 && year % 100 != 0))
         return 29;
      else
         return 28;
   }
   //months which has 31 days
   else if (month == 1 || month == 3 || month == 5 || month == 7 || month == 8
      || month == 10 || month == 12)
      return 31;
   else
      return 30;
```

```
}
```

Question 3)

```
// using the NumberOfDays function above calculate the number of days
// between two dates
// DiffDays(2020, 7, 19,2020, 7, 19) ==> 0
// DiffDays(2020, 7, 19,2020, 7, 20) ==> 1
// DiffDays(2020, 7, 19,2020, 8, 19) ==> 31
int DiffDays(int startYear, int startMonth, int startDay,
   int endYear, int endMonth, int endDay)
{
   int numOfDays = 0;

   while (true)
   {
      if (startYear == endYear &&startMonth == endMonth &&
         startDay == endDay)
         break;

      numOfDays++;
      if (startDay == NumberOfDaysInMonth(startMonth, startYear))
      {
         startDay = 1;
         startMonth++;
      }
      else
         startDay++;

      if (startMonth >= 12)
      {
         startMonth = 1;
         startYear++;
         startDay = 1;
      }
   }

   return(numOfDays);
}
```