Problem Set 1 Purpose: Introduction to iterative statement, write program using for/while.

Problem 1:
How many times will the below loop run?

```
#include<iostream>

using namespace std;


void main()

{

        int count = 1;

        int x=10,y=20;


        while(x<y)//in parenthesis we have condition, which when true takes us inside loop

        {

                x = x+1;

                y = y-1;

                cout<<"Iteration "<<count<<endl;

                count++;

        }


}
```

Note: *Notice the indentation (spacing) in the code above. Indentation makes the code readable.*

Problem 2:
Write a program which keeps taking input from user until he enters -1. On receiving -1 your program will display average of numbers entered.

Problem 3:
A number *x* can be determined to be a prime number by a method called trial division. Check if *x* can be divided by 2,3,4,…sqrt(*x*). If it can be divided then it is not a prime number otherwise it is. For example, to check if 29 is a prime number, take the square root of 29 and round it to the nearest integer, i.e., 5.3852 would be 6. Now check if 29 is divisible by 2,3,4,5 or 6. As it cannot be divided by any of these numbers it is a prime number. Your program should exit the loop as soon as it finds the first divisor.

Problem 4:
Input a number from the user and output its digits, one digit per line. For example, if the user inputs 7854, then your program should output:

```
The digits of 7854 are:
```

```
7

8

5

4
```

Note: Your program should run for any number of digits supported by an "int" (i.e. from 0 to max number). Modify your program to print invert of the number. For example, invert of 7854 is 4587.

Problem 5:

Input 10 numbers from the user and print the sum of all numbers which are greater than 10.

Problem 6:

Write a program which takes a number 'n' from user and displays its factorial.

Problem 7:

Keep inputting two numbers from the user. You have to stop when the sum of the two numbers is odd; otherwise input the two numbers again. At the end you have to output the product of all the numbers input by the user.

Problem 8:

Determine the pattern followed by the series below:

1, 2, 3, 5, 8, 13, 21, 34, …

Write a program that takes a number 'n' and prints first 'n' numbers of this series. For example, if user enters 6, your program should display

1, 2, 3, 5, 8, 13

Problem 9:

Write a C++ program that takes 2 integers (i) a starting number and (ii) the number of elements in last row. Then print then prints the data in form a lower triangular matrix. For example, if the starting number is 10 and the number of elements in last row is 3, then output would be

```
10
11  12
13 14 15
```

Note: You have to do it using a single while loop.

Problem 10:

An Armstrong number of three digits is an integer such that the sum of the cubes of its digits is equal to the number itself. For example, 371 is an Armstrong number since 3**3 + 7**3 + 1**3 = 371.Write a program to find all Armstrong number in the range of 0 and 999.

Problem 11:

Write a C++ program that keeps on taking integers until -999 is entered. The program then displays the sum of all integers, the largest and the smallest integers.

Problem 12:

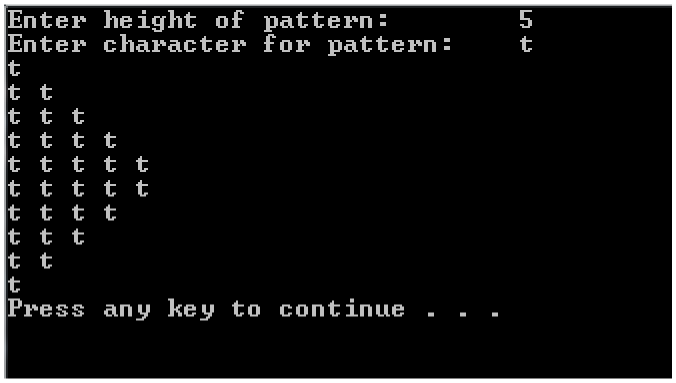
Write a program to calculate the place value of digit in an integer.

For example, if the user inputs an integer *56918* and you want to determine the place value of *6*, the output would be "Thousands". You can include a check for whether that specific digit is present or not.Your program should work for a maximum place value of "millions".

**Units; tens; hundreds; thousands; ten thousands; hundred thousands; millions.*

Problem 13:

Write a program that prints a triangular pattern. The program asks the user to enter the height of the triangle and the character to use. For example:

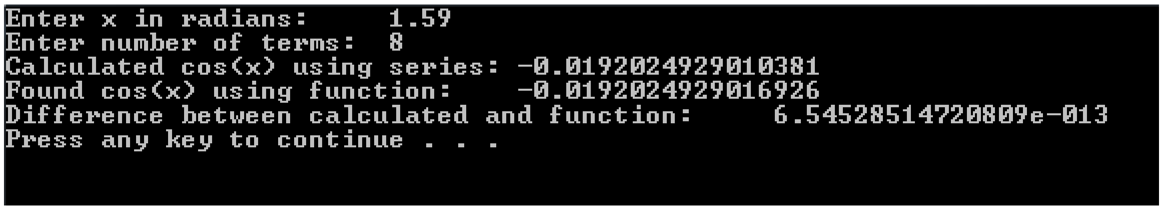


```
Enter height of pattern:        5
Enter character for pattern:    t
t
t t
t t t
t t t t
t t t t t
t t t t t
t t t t
t t t
t t
t
Press any key to continue . . .
```

Problem 14:
The trigonometric series to find cos(x), where x is a number in radians, is given as:

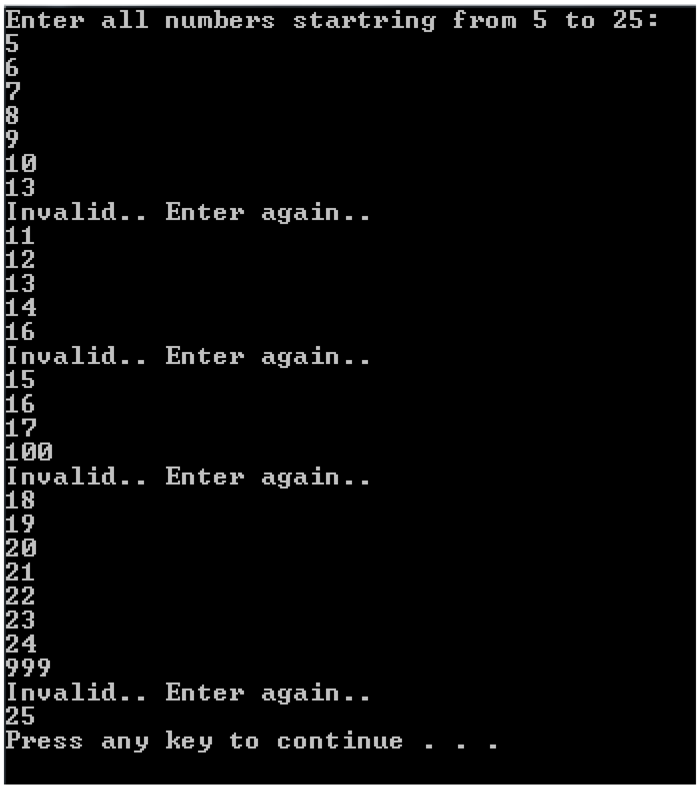
$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$$

Write a program that asks the user for x and the number of terms to use to find cos(x). Find the value of cos(x) using available function and compute the difference between the calculated value (using your program) and the value found from the function. Answer should be displayed with an accuracy of 15 decimal places. For example:



```
Enter x in radians:     1.59
Enter number of terms:  8
Calculated cos(x) using series: -0.0192024929010381
Found cos(x) using function:    -0.0192024929016926
Difference between calculated and function:     6.54528514720809e-013
Press any key to continue . . .
```

Problem 15:
Write a program in which the user has to enter all numbers between 5 and 25 (inclusive). If the user does not enter the correct number, display an error and ask to enter again. For example:



```
Enter all numbers startring from 5 to 25:
5
6
7
8
9
10
13
Invalid.. Enter again..
11
12
13
14
16
Invalid.. Enter again..
15
16
17
100
Invalid.. Enter again..
18
19
20
21
22
23
24
999
Invalid.. Enter again..
25
Press any key to continue . . .
```

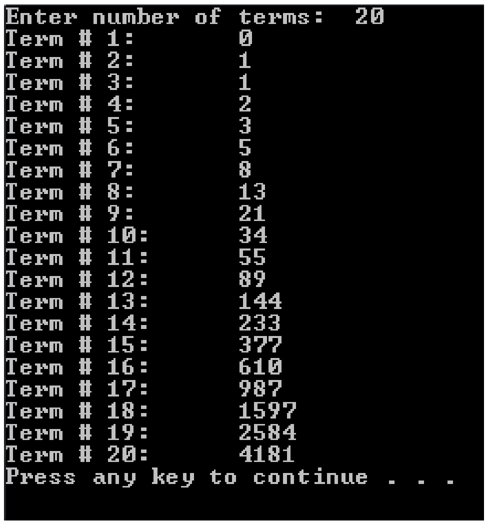
Problem 16:
π can be calculated using the following infinite series:

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \ldots$$

Write a code that prints the value of π after the first 100 terms, 200 terms and 300 terms (in the same code).

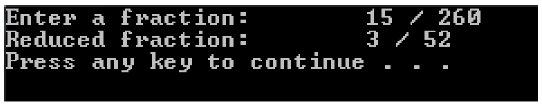
Problem 17:
Write a code that finds out if a number is a prime number. The user enters a number and your code must output whether it is a prime number or not.

Problem 18:
Fibonacci sequence is calculated such that the n^{th} term is the sum of $n – 1^{th}$ and $n – 2^{th}$ term. The first number in the sequence is 0 and the second number is 1. Therefore, the third number would be 0 + 1 = 1, the fourth number would be 1 + 1 = 2, the seventh number would be 5 + 8 = 13, and so on. Write a program that asks the user to enter a number (representing the number of terms to be found). The program should write all the numbers in the sequence up to, and including, this term. For example,



```
Enter number of terms:  20
Term # 1:       0
Term # 2:       1
Term # 3:       1
Term # 4:       2
Term # 5:       3
Term # 6:       5
Term # 7:       8
Term # 8:       13
Term # 9:       21
Term # 10:      34
Term # 11:      55
Term # 12:      89
Term # 13:      144
Term # 14:      233
Term # 15:      377
Term # 16:      610
Term # 17:      987
Term # 18:      1597
Term # 19:      2584
Term # 20:      4181
Press any key to continue . . .
```

Problem 19:
Write a program that takes in a fraction and then displays that fraction in the reduced form. For example:



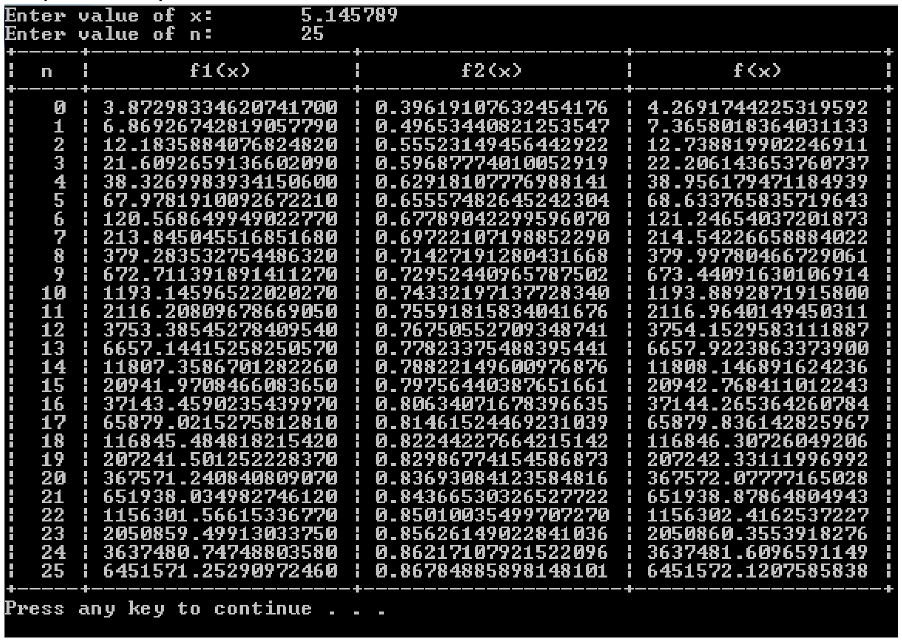
```
Enter a fraction:           15 / 260
Reduced fraction:            3 / 52
Press any key to continue . . .
```

Problem 20:
Write a program that calculates:

$$f(x)=f1(x)+f2(x),$$

where $f1(x)=\sqrt{15*(x-2)}^{n}$ and $f2(x)=\log_{10}$¿ .

The user will enter the value of x and n (where n is the number of terms to be calculated, starting from $n=0$). The output should be formatted as below:

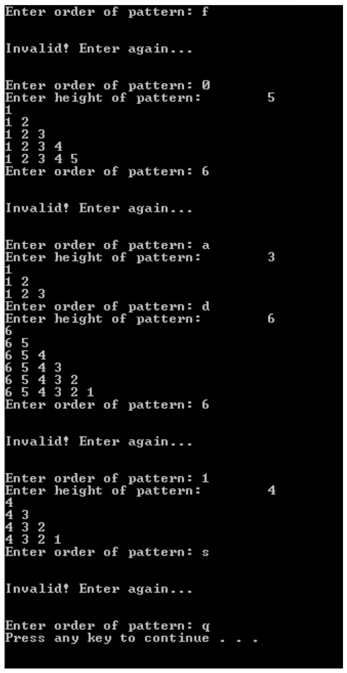


```
Enter value of x:       5.145789
Enter value of n:       25
+------+----------------------+----------------------+----------------------+
¦  n   ¦        f1(x)         ¦        f2(x)         ¦         f(x)         ¦
+------+----------------------+----------------------+----------------------+
¦    0 ¦ 3.872983346207417000 ¦ 0.396191076324541760 ¦ 4.2691744225319592   ¦
¦    1 ¦ 6.869267428190577900 ¦ 0.496534408212535470 ¦ 7.3658018364031133   ¦
¦    2 ¦ 12.18358840768248200 ¦ 0.555231494564429220 ¦ 12.738819902246911   ¦
¦    3 ¦ 21.60926591366020900 ¦ 0.596877740100529190 ¦ 22.206143653760737   ¦
¦    4 ¦ 38.32699839341506000 ¦ 0.629181077769881410 ¦ 38.956179471184939   ¦
¦    5 ¦ 67.97819100926722100 ¦ 0.655574826452423040 ¦ 68.633765835719643   ¦
¦    6 ¦ 120.5686499490227700 ¦ 0.677890422995960700 ¦ 121.24654037201873   ¦
¦    7 ¦ 213.8450455168516800 ¦ 0.697221071988522900 ¦ 214.54226658884022   ¦
¦    8 ¦ 379.2835327544863200 ¦ 0.714271912804316680 ¦ 379.99780466729061   ¦
¦    9 ¦ 672.7113918914112700 ¦ 0.729524409657875020 ¦ 673.44091630106914   ¦
¦   10 ¦ 1193.145965220202700 ¦ 0.743321971377283400 ¦ 1193.8892871915800   ¦
¦   11 ¦ 2116.208096786690500 ¦ 0.755918158340416760 ¦ 2116.9640149450311   ¦
¦   12 ¦ 3753.385452784095400 ¦ 0.767505527093487410 ¦ 3754.1529583111887   ¦
¦   13 ¦ 6657.144152582505700 ¦ 0.778233754883954410 ¦ 6657.9223863373900   ¦
¦   14 ¦ 11807.35867012822600 ¦ 0.788221496009768760 ¦ 11808.146891624236   ¦
¦   15 ¦ 20941.97084660836500 ¦ 0.797564403876516610 ¦ 20942.768411012243   ¦
¦   16 ¦ 37143.45902354399700 ¦ 0.806340716783966350 ¦ 37144.265364260784   ¦
¦   17 ¦ 65879.02152758128100 ¦ 0.814615244692310390 ¦ 65879.836142825967   ¦
¦   18 ¦ 116845.4848182154200 ¦ 0.822442276642151420 ¦ 116846.30726049206   ¦
¦   19 ¦ 207241.5012522283700 ¦ 0.829867741545868730 ¦ 207242.33111996992   ¦
¦   20 ¦ 367571.2408408090700 ¦ 0.836930841235848160 ¦ 367572.07777165028   ¦
¦   21 ¦ 651938.0349827461200 ¦ 0.843665303265277220 ¦ 651938.87864804943   ¦
¦   22 ¦ 1156301.566153367700 ¦ 0.850100354997072700 ¦ 1156302.4162537227   ¦
¦   23 ¦ 2050859.499130337500 ¦ 0.856261490228410360 ¦ 2050860.3553918276   ¦
¦   24 ¦ 3637480.747488035800 ¦ 0.862171079215220960 ¦ 3637481.6096591149   ¦
¦   25 ¦ 6451571.252909724600 ¦ 0.867848858981481010 ¦ 6451572.1207585838   ¦
+------+----------------------+----------------------+----------------------+
Press any key to continue . . .
```

(HINT: Look up functions available under iomanip and iostream libraries)

Problem 21:

Write a program that prints a triangular pattern. The program asks the user for the height of the triangle and whether to print the triangle in ascending or descending order. The user must enter either 0 or a (for ascending pattern) and either 1 or d (for descending pattern). If the user enters an invalid option, the program must display an error and ask for input again. Program should terminate if the user enters the character q. For example:



```
Enter order of pattern: f


Invalid! Enter again...


Enter order of pattern: 0
Enter height of pattern:        5
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
Enter order of pattern: 6


Invalid! Enter again...


Enter order of pattern: a
Enter height of pattern:        3
1
1 2
1 2 3
Enter order of pattern: d
Enter height of pattern:        6
6
6 5
6 5 4
6 5 4 3
6 5 4 3 2
6 5 4 3 2 1
Enter order of pattern: 6


Invalid! Enter again...


Enter order of pattern: 1
Enter height of pattern:        4
4
4 3
4 3 2
4 3 2 1
Enter order of pattern: s


Invalid! Enter again...


Enter order of pattern: q
Press any key to continue . . .
```

Problem 22:
Write a C++ program that keeps on taking integers until -999 is entered.
The program then displays the sum of all integers, the largest and the smallest integers.
Note: -999 would not be included in the calculations.
Modify your code so that it now tells the total number of integers entered before -999, and at what positions the smallest and largest integers were entered. Exceeds

Problem 23:
The population of a town A is less than the population of town B. However, the population of town A is growing faster than the population of town B. Write a program that prompts the user to enter the population and growth rate of each town. The program outputs after how many years the population of town A will be greater than or equal to the population of town B and the populations of both the towns at that time. (A sample input is: Population of town A = 5000, growth rate of town A = 4%, population of town B = 8000, and growth rate of town B = 2%.)

Problem 24:
Write a program to calculate the place value of digit in an integer. Your program would take 2 integers from the user, an integer between 0 and 10 million, and another integer – a single digit in the first number. Your program would then output the place value of that digit in the number.
For example, if the user inputs an integer *56918* and you want to determine the place value of *6*, the output would be “Thousands”.
You can include a check for whether that specific digit is present or not.
**Units; tens; hundreds; thousands; ten thousands; hundred thousand; millions.*

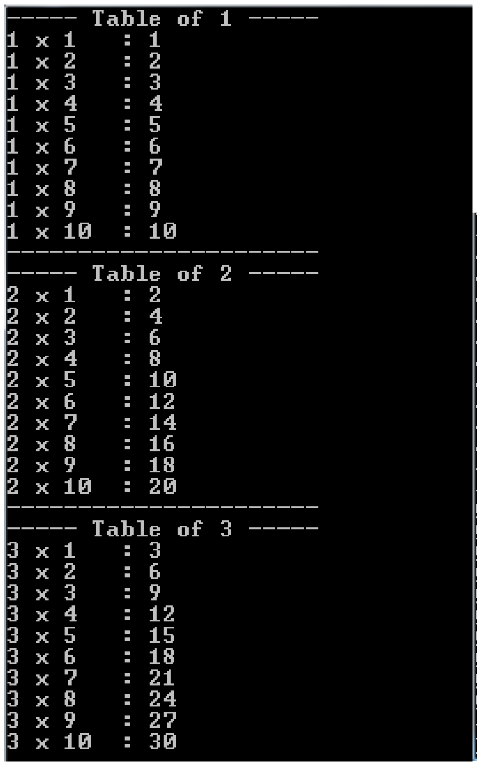
Problem 25:

Design and write a C++ program that takes as input an integer larger than 1 and calculates the sum of the squares from 1 to that integer. The output should be the value of the squares and the sum, properly labelled on the screen.
For example, if the integer equals 4, your program would display “1 + 4 + 9 + 16 = 30”

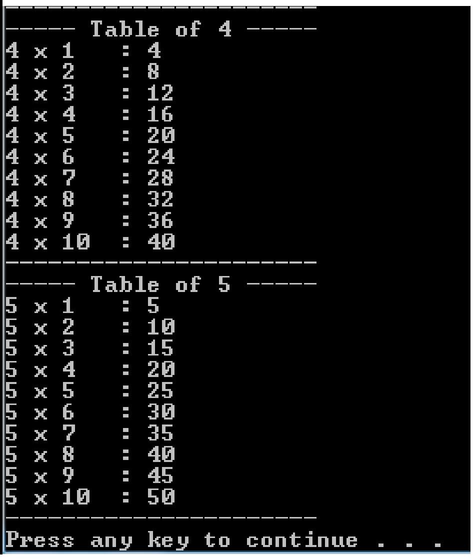
Problem 26:
Suppose that the first number of a sequence is x, in which x is an integer. Define a_0 = x; a_{n+1} = a_n/2 if a_n is even; a_{n+1} = 3 x a_n + 1 if a_n is odd. Then, there exists an integer k such that a_k= 1. Write a program that prompts the user to input the value of x. The program output the integer k such that a_0 = 1 and the numbers,$a_0, a_1, a_2, \ldots, a_k$. (For example, if x = 75, then k = 14, and the numbers, $a_0, a_1, a_2, \ldots, a_{14}$, respectively, are 75, 226, 113, 340, 170, 85, 256, 128, 64, 32, 16, 8, 4, 2, 1.) Test your program for the following values of x: 75, 111, 678, 732, 873, 2048, and 65535.

Problem Set 2 Purpose: Introduction to nested loops, write program using while ()/for (;;) in while ()/for (;;).

Problem 1: Write a program that uses nested for – loops to print first 10 entries of tables of 1 to 5. For example:



```
----- Table of 1 -----
1 x 1   : 1
1 x 2   : 2
1 x 3   : 3
1 x 4   : 4
1 x 5   : 5
1 x 6   : 6
1 x 7   : 7
1 x 8   : 8
1 x 9   : 9
1 x 10  : 10
----------------------
----- Table of 2 -----
2 x 1   : 2
2 x 2   : 4
2 x 3   : 6
2 x 4   : 8
2 x 5   : 10
2 x 6   : 12
2 x 7   : 14
2 x 8   : 16
2 x 9   : 18
2 x 10  : 20
----------------------
----- Table of 3 -----
3 x 1   : 3
3 x 2   : 6
3 x 3   : 9
3 x 4   : 12
3 x 5   : 15
3 x 6   : 18
3 x 7   : 21
3 x 8   : 24
3 x 9   : 27
3 x 10  : 30
```



```
----------------------
----- Table of 4 -----
4 x 1   : 4
4 x 2   : 8
4 x 3   : 12
4 x 4   : 16
4 x 5   : 20
4 x 6   : 24
4 x 7   : 28
4 x 8   : 32
4 x 9   : 36
4 x 10  : 40
----------------------
----- Table of 5 -----
5 x 1   : 5
5 x 2   : 10
5 x 3   : 15
5 x 4   : 20
5 x 5   : 25
5 x 6   : 30
5 x 7   : 35
5 x 8   : 40
5 x 9   : 45
5 x 10  : 50
----------------------
Press any key to continue . . .
```

Problem 2: Find all prime numbers from 1 to a user defined number, N.

Problem 5: Every single part of this question has something to do with prime numbers.

I. Write a C++ program that takes any 2-digit number and determines whether that number is a prime or not.

II. Now consider your roll number, take the last four digits of your roll number and calculate $Roll_{no}^{0.8}$, e.g. if your roll-no is L14-5134, you would calculate $5134^{0.8}$. Determine the prime number before and after the number $Roll_{no}^{0.8}$.
You may call the former as prime_former and the latter as prime_latter.
A number, limit, is equal to prime_former if $|Roll_{no}^{0.8}$ – prime_former| is greater than | $Roll_{no}^{0.8}$ – prime_latter|, else limit is equal to prime_latter; where |x| represents the magnitude of the quantity x.
You have to calculate and display all the prime numbers between the 2-digit number from part (I) and the number limit.

III. Modify your program from part (II), such that it only displays all the prime numbers that have the digit 7 in them.

Problem 6: Write a C++ program that displays the first 15 decimal numbers, along with their binary equivalents.
The output of your program would be something like,

BIT-1	BIT-2	BIT-3	BIT-4	DECIMAL EQUIVALENT
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Problem 7: Write a C++ program that uses nested loops to output the following pattern. The height of the pattern is to be input by the user and should not be greater than 9.

```
1 2 3 4 5 6 7 6 5 4 3 2 1
1 2 3 4 5 6   6 5 4 3 2 1
1 2 3 4 5       5 4 3 2 1
1 2 3 4           4 3 2 1
1 2 3               3 2 1
1 2                   2 1
1                       1
```

Problem 8: Problem 19-29 in purpose 11.