Introduction to Programming
Practice Problems-2

Welcome back :-) Let us move to repetitive execution of the statement(s) with the help of loops in today's lab. Following is the syntax of ***while*** loop in C.

```
while (expression) {
    statements;
}
```

The statements in the block executes till the ***expression*** evaluates to true. for example:

```
i=0;
while (i<=10) {
    printf("%d\n",i);
    i++;
}
```

The above loop will print numbers from 0-10. There are three main components of every loop: *initialization* of the loop index (i=0 in our example), loop controlling *condition* ($i <= 10$) and *updation of loop index* inside the loop which is $i++$ in our sample example. It is important to move towards the terminating condition of the loop after each iteration. We will study other loops like **for**, **do-while** in this weeks classes. Try to develop programs for the following sample problems. If you have any doubt then consult you TA's. All the best.

A few new problems on loops

1. Enter an integer number, find and print all the prime numbers till that number.
2. Write programs to print the following patterns. Print only one $*$ at a time with the printf statement.

   (a)
   ```
   *
   * *
   * * *
   * * * *
   * * * * *
   ```

   (b)
   ```
         *
       * *
     * * *
   * * * *
   * * * * *
   ```

3. Write a program to print Floyd's triangle as shown below with single and dual loops.

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

4. Find the GCD of two nunbers.

5. Write a program to print the n^{th} term of e^x series. Take x and n as input. Verify your answers using calculator.

6. Write a program to find the $sin(x)$ and $cos(x)$ till 4 decimal places. Take x as input. Verify your answers using calculator.

7. Generate all r-combinatorions of n elements when repetitions are allowed. For example, here are all 3-combinations of 5 elements:

```
111 112 113
114 115 122
123 124 125
133 134 135
144 145 155
222 223 224
225 233 234
235 244 245
255 333 334
335 344 345
355 444 445
455 555
```

8. Write a program to play tic tac toe game with computer as one opponent. Randomly select moves of the computer using the *rand* and *srand* function declared in **"stdlib.h"**

Problems from previous lab

1. Read two integers and print numbers from the lower integer to the higher.

```
Input:
   10
   6
Output:
   6
   7
   8
   9
   10
```

2. Read n real numbers and print the maximum. First input is n. Eg:

```
Input:
   5
   1.0
   10.2
   2.4
   -4.2
   2.5
Output:
   10.2
```

3. Find average of n numbers.

```
Input:
   3
   0.7
   0.3
   2.5
Output:
   1.1667
```

4. Write a program which takes an integer as input and uses the Babylonian method to compute its square root. Your program should check for the case when the input integer is negative.

   The following is the pseudo-code of ***Babylonian*** method

```
1 Start with an arbitrary positive start value x (the closer to the
   root, the better).
2 Initialize y = 1.
3. Do following until desired approximation (accuracy to the
  required number of decimal places) is achieved.
  a) Get the next approximation for root using average of x and y
  b) Set y = n/x
```

5. In this exercise, we will compute some parameters of the motion of a projectile thrown from a height h with velocity v at an angle θ to the x-axis. The program will be given the three numbers: height h, velocity v, and angle θ. The program should output:

   - The maximum height reached by the projectile.
   - The distance travelled by the projectile in the x-direction before it hits the ground.
   - The speed of the projectile when it hits the ground.

6. Check whether a number is armstrong or not. A number is armstrong if the sum of cubes of individual digits of a number is equal to the number itself. For example 371 is an armstrong number as $27 + 3443 + 1 = 371$.

7. Write a program to display first n numbers of the fibonacci sequence: $0, 1, 1, 2, 3 \cdots$. Take n as an integer input.