P.1. Write a program to generate the following patterns.

```
1
12
123
1234
```

```
   *
  ***
 *****
*******
```

```
1
23
456
78910
```

Ans.

*Algorithm:

1) Declare i and j as integer and read the value of n.

2) In a 'for' loop, initialize value i ← 1 is declared, and repeat the steps until i<=n and i ← i+1

3) Again, in a ~~loop~~ 'for' loop, initialize value j ← 1 is declared, and repeat the steps until j<=i and j ← j+1

4) Print the value of i.

5) Print a new line.

*Program Code:

```
#include<stdio.h>
  Void main()
      {
        int i=1,j=1;
```

```
printf("Enter the number of rows: ");
scanf("%d", &n);
for(i=1, i<=n; i++)
  {
   for(j=1; j<=i; j++)
     {
      printf("%d\t", j);
     }
    printf("\n");
  }
 return 0;
}
```

* Input:

Enter the number of rows: 4

* Output:

```
1
1  2
1  2  3
1  2  3  4
```

* Algorithm:

1) Declare i and j,
2) Initiative value i ← 1.
3) Repeat the steps i<=4 and i ← i+1
4) Again initiative value j ← 1.
5) Repeat the steps j<=i+3 and i ← i+1
6) Test whether i+j<=4
7) Then print " "
8) Otherwise, print "*"
9) Print "\n"

* Program Code:

```
#include<stdio.h>
  void main()
      {
       int i,j;
       for(i=1; i<=4; i++)
          {
           for (j=1, j<=i+3; j++)
              {
               if(i+j<=4)
                 printf(" ");
```

```
            else  else
                printf("*");
            }
        printf("\n");
    }
    return 0;
}
```

* Output:

```
        *
      * * *
    * * * * *
  * * * * * * *
```

* Algorithm:

1) Declare ~~ian~~ i, j and c as integer variable.
2) In a 'for' loop, initialize value i ← 1 is declared, and repeat the steps until i <= 4, and i ← i+1.
3) Again, in a 'for' loop, initialize value j ← 1 is declared, and repeat the steps until j <= i and j ← j+1

4) Initialize $c \leftarrow 1$, print the value of c, $c = c+1$.

5) Print a new line.

* Program Code:

```
#include<stdio.h>
  void main()
      {
        int i,j, c=1;
        for (i=1; i<=4; i++)
           {
            for(j=1; j<=i; j++)
               {
                printf("%d", c);
                c++;
               }
            printf("\n");
           }
      }
```

* Output:

```
1
2 3
4 5 6
7 8 9 10
```

* Discussion:

These programs satisfy CO1 and CO3.

P.42. Write a program to generate ~~all prime~~ the non-Fibonacci numbers up to a given limit (number) provided by user.

Ans.

* Algorithm:

1) Declare n, a, b, c, d and x and read n.
2) Initialize value of a ← 0, b ← 1 and c ← 0.
3) Repeat the steps until c <= n.
   3.1. c ← a+b
   3.2. a ← b
   3.3. b ← c
   3.4. d ← a+b
4) Initialize value of x ← c+1.
5) Repeat the steps until x < d and x ← x+1.
6) Test whether x <= n or not.
7) Then print the value of x.
8) Otherwise break.

* Program Code:

```
#include<stdio.h>
   Void main()
        {
          int n,i,a,b,c,d,x;
```

```
a=0;
b=1;
c=0;
printf("Enter the upper range :");
scanf("%d",&n);
printf("Non-Fibonacci series is:\n");
While(c<=n)
    {
      c=a+b;
      a=b;
      b=c;
      d=a+b;
      for(x=c+1;x<d;x++)
        {
          if(x<=n)
            printf("%d\t",x);
          else
            break;
        }
    }
}
```

* Input:

Enter the upper range : 10

* Output:

Non-Fibonacci series is

4 6 7 9 10

* Discussion:

This program satisfies CO1 and CO3.

Q3. Write a program to generate find the sum of the following series connect upto a given number of decimals provided by the user:

$$\frac{2}{1.3} + \frac{4}{3.5} + \frac{6}{5.7} + \cdots$$

Ans.

* Algorithm:

1) Read a, n
2) Take float variable, p = 0.0, term = 0.0, sum = 0.0.
3) Take a 'for' loop which starts from, 1 and run until i<=n value of i is incremented by 1.
4) Then 2 * i
5) term ← p/((p-1)*(p+1));
6) Sum ← sum + term
7) print sum.

*Program Code:

```
#include<stdio.h>
  void main()
       {
        int n,a,i=0;
        printf("Enter the limit of the
                        series:");
        scanf("%d",&n);
        printf("Enter the number of
                    decimal places:");
        scanf("%d",&a);
        float p=0.0;term=0.0,sum=0.0;
        for(i=1;i<=n;i++)
           {
            p=2+i;
            term=p/((p-1)*(p+1));
            sum=sum+term;
           }
         printf("The sum of the series
                  correct upto %d decimal
                  places is %f",a,sum);
  }
```

* Input:

Enter the limit of the series : 2

Enter the numben of decimal places : 2

* Out put:

The sum of the series connect upto 2 decimal places is 0.93

* Discussion:

The program satisfies CO1 and CO3.

P.4. Write a program to generate all prime numbers up to a given number.

Ans.

* Algorithm:

1) Declare i and j and read n.
2) Initialize value c ← 0, i ← 1
3) Repeat the steps until i <= n and i ← i+1
4) Initialize value j ← 1
5) Repeat the steps until j <= i and j ← j+1
6) c ← c+1
7) Print the value of c.
8) Print "\n"

* Program Code:

```
# include <stdio.h>.
   Void main()
        {
         int n,i, fact,j;
         printf("Enter the number : ");
         scanf ("%d", &n);
         printf("Prime numbers are: \n");
```

```
for (i=1; i<=n; i++)
    {
     for(j=1; j<n ; j++)
        {
         if (i%j=0)
         fact++;
        }
      if (fact==2)
        printf("%d\t",i);
    }
  return 0;
}
```

* Input :

  Enter the number : 10

* Output :

  Prime numbers are :

  2 3 5 7

* Discussion :

  This program satisfies CO1 and CO3.

P.5. Write a program to generate all the Armstrong numbers from 100 to 1000.

Ans.

* Algorithm:

1) Generate a loop from 100 to 1000
2) Extract the digit from the numbers.
3) Do the cube of the digit and check if the sum of the cube is equal to the given numbers.
4) Print the Armstrong numbers.

* Program Code:

```
#include<stdio.h>
   void main()
        {
          int i,j,n,c=0;
          for(i=100; i<1000; i++)
             {
               for(j=1; j<=3; j++)
                  {
                    n=i%10;
```

```
            c=c+n*n*n;
            i=i/10;
        }
    if(c==i)
      printf("%d",i);
  }
}
```

*Output:

Armstrong number of 3 digit is

$abc = a^3 + b^3 + c^3$

$= 1^3 + 5^3 + 3^3$

$= 153$

*Discussion:

This program satisfies CO1 and CO3.

Questionnaire:

1. Explain the difference between *break* and *continue*.
2. What will happen if a *break* statement is encountered within a nested loop? Explain with an example.
3. Explain with an example how the value of the loop variable of an outer loop may control the number of iterations of an inner loop. Can the value of the loop variable of an inner loop control the number of iterations of an outer loop?

Answer;

1. If 'break' statement is encounted, control will go out of the loop and will never enter the loop again.

   If 'continue' is encountered, then the statements present after it will not be executed and control will enter the next iteration of the loop.

2. 'Break' statement in the inner loop of a nested loop means the inner loop will not be further executed.

Ex.
```
int num = 20;
for (int i = 1; i <= 2; i++)
  {for (int y = 1; y <= 50; y++)
    {if (j == num)
      break;
    } }
  } }
```

3.
```
for (i=1; i<=20; i++)
  {for (j = 2; j <= i, j++)
    {if (i % j == 0)
      break;
    }
  }
```

No, the loop variable of the inner loop cannot control the number of iterations of the outer loop.

Grade awarded:	Teacher's signature with date :