Dhaka International University

Knowledge is Power
DIU
Estd. 7th April, 1995

DEPARTMENT OF CSE

LAB REPORT

COURSE NAME : Structured Programming language Lab

COURSE CODE : 0613-102

REPORT NO : 09

REPORT ON : Prime, co-prime or not check and some program in C programming.

SUBMITTED BY	SUBMITTED TO
NAME : Mir Yeasir Abrar **ROLL** : 35 **REG. NO** : CS-0-98-23-127358 **BATCH** : 98 (1st shift) **SEMESTER** : 1st	Rakib Mahmud Lecturer Dept of CSE

DATE OF SUBMISSION : 23 November 2024

Date of Performance: 16 Novembe 2024

Title: Prime, co-prime or not check & some Programs in C Programming.

Objective:

- To implement a program that checks wheather a number is prime and whether the numbers are co-prime or not.
- To write a program to find prime number from 1 to 100 and another program to check if a number is strong or not using loops and mathematical logic
- To explore 8 different pattern program using loops.

Introduction:

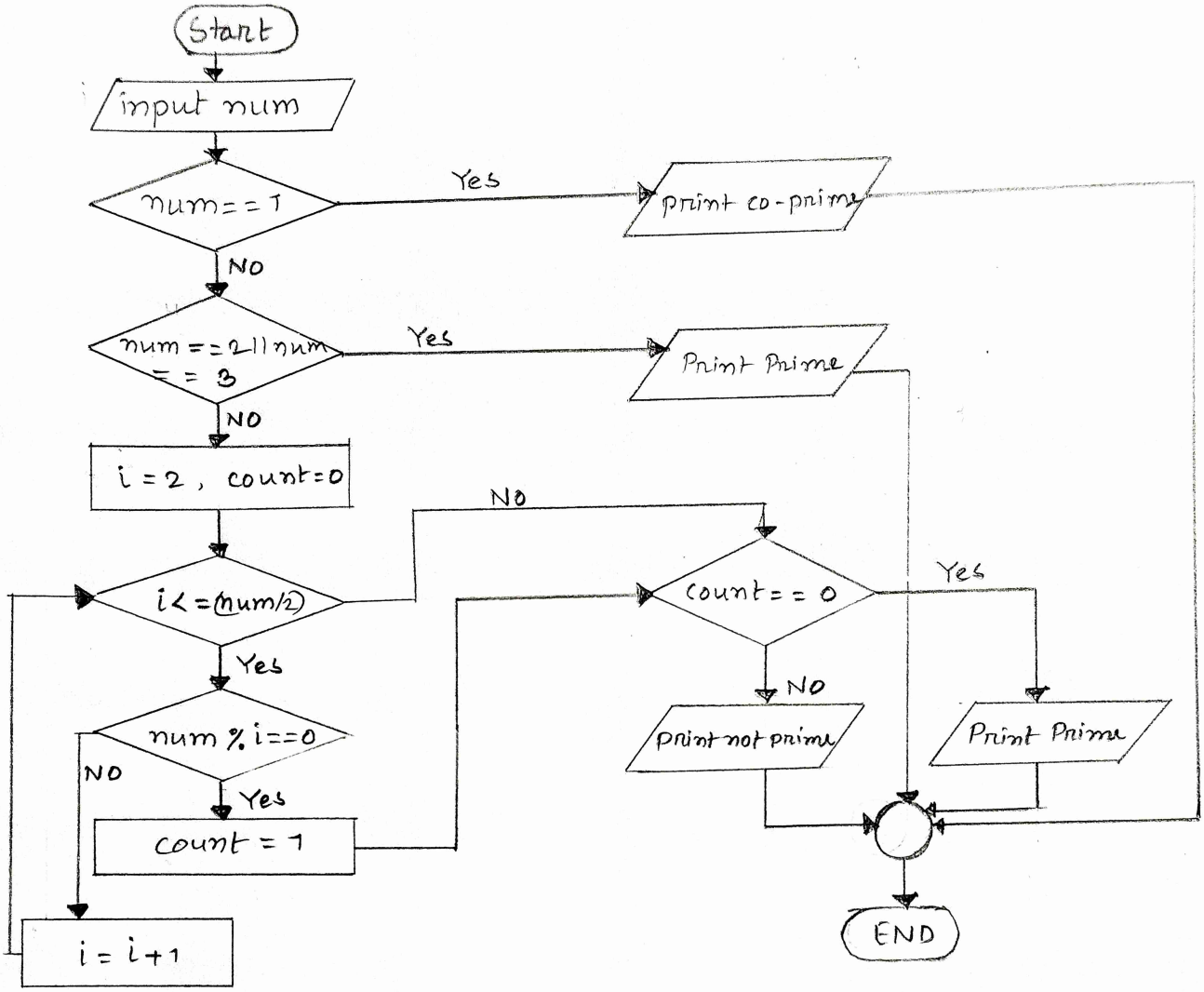
This lab focuses on basic number manipulations and patterns using loops, conditional statements and arithmatic operations in C programming.

Prime Number Check

... Explanation: Prime Numbers are numbers greater than 1 that are divisible only by one and themselves. The provided program checks wheather a given number is a prime by evaluating its divisibility by numbers from 2 up to half of the given number. It also includes edge cases for small numbers such as 1, 2 and 3.

- co-prime check: Numbers such as 1 are considered co-prime with any other number since their greatest common divisor (GCD) with any number is 1.
- The program utilize conditional statements and loop to determine primality.

Flowchart:



Algorithm:

01. start
02. Input num
03. i=2, count = 0
04. Is (num==1)
    - Yes, print co-prime and go to step 9
    - No, go to next step
05. Is (num==2) OR (num==3)
    - Yes, print prime and go to step 9
    - No, go to next step
06. Is i<=(num/2)
    - Yes, go to step 7

- No, go to step 8

07. Is (num % i == 0)
    - Yes, count = 1, go to step 8
    - No, go to step 6

08. Is (count == 0)
    - Yes, print prime number
    - No, print not prime number

09. End.

Discussion:

In this section we will learn about presentation of code and out-put of code. The shown code will be as in IDE and the output will be as in console.

01. check prime, co-prime or not program

...code:

```
# include <stdio.h>
 int main (){
    int num, count = 0, i, j;
    printf (" Enter any positive number : ");
    scanf (" %d ", &num);

    if (num == 1)
   } printf (" %d is a co-prime number \n ", num);
    }
    else if (num == 2 || num == 3)
    {
      printf (" %d is a prime number \n ", num);
    }
```

```
    else
    {   for (i=2; i<=(num/2); i++)
        {
            if (num % i == 0)
            { count = 1;
              break;
            }
        }
        if (count == 1)
        { printf(" %d is not prime number\n", num);
          break;
        }
        else
        { printf(" %d is a prime number \n", num);
          break;
        }

    }
}
```

... Output:

```
Enter any positive number: 13
13 is a prime number
```

02. Find prime numbers from 1 to 100 program

... Code:

```
#include <stdio.h>
int main()
{   int i, num, count;
    for (num = 1; num <= 100; num++)
    {
        count=0;
```

```
        for (i=2; i<= num/2; i++)
        {
            if (num % i == 0)
            {
                count++;
                break;
            }
        }

        if (count == 0 && num != 1)
        {
            printf("%d \t", num);
        }
    }
    return 0;
}
```

... Output:

```
 2   3   5   7   11   13   17   19   23   29   31   37   41
43   47   53   59   61   67   71   73   79   83   89   97
```

03. A c program to check if a number is strong or not.

... code:

```
#include <stdio.h>
int main()
{
    int num, temp, rem, count, fact, sum=0;
    printf("Enter a number : ");
    scanf("%d", &num);
    temp = num;
```

```
        while (num)
        {
            rem = num % 10;

            count = 1;
            fact = 1;

            while (count <= rem)
            {
                fact = fact * count;
                count++;
            }

            printf("Factorial of %d is %d \n", rem, fact);

            sum = sum + fact;

            num = num / 10;
        }

        if(temp == sum)
        {
            printf("%d is a strong number \n", temp);
        }
        else
        { printf("%d is not a strong number \n", temp);
        }

        return 0;
}
```

... output:

```
Enter a number: 145
Factorial of 5 is 120
Factorial of 4 is 24
Factorial of 1 is 1
145 is a strong number
```

04. Pattern type 01 -

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

... code:

```
#include <stdio.h>
int main()
{
    int n;
    printf("Enter the value of n : ");
    scanf("%d", &n);

    for(int row=1 ; row <=n ; row++)
    {
        for(int col = 1 ; col <= row ; col++)
        {
            printf("%d ", col);
        }
        printf("\n");
    }
}
```

... output:

```
Enter the value of n : 5
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

05. Pattern type 02 -

```
1
1 0
1 0 1
```

... code:

```
#include <stdio.h>
int main()
{
```

```
    int n;
    printf("Enter the value of n: ");
    scanf("%d", &n);

    for (int row=1; row <= n; row++)
    {
       for(int col = 1; col <= row; col++)
        if (col %2 == 0)
          { printf("0 ");
          }
         else
          { printf("1 ");
          }
          printf("\n");
      }
}
```

... output:

```
Enter the value of n: 5
1
1 0
1 0 1
1 0 1 0
1 0 1 0 1
```

06. Pattern type -03

```
A
A B
A B C
```

... code:

```
#include <stdio.h>
int main ()
{
   int n;
   printf("Enter the value of n: ");
```

```
        scanf("%d", &n);
        for (int row = 1; row <= n; row++)
        {
            for (int col = 1; col <= row; col++)
            { printf("%c", col+64);
            }
            printf("\n");
        }
}
```

... Output:

```
Enter the value of n: 5
A
A B
A B C
A B C D
A B C D E
```

07. Pattern type : 04 -

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

... Code:

```
# include <stdio.h>
int main()
{
    int n;
    printf("Enter the value of n : ");
    scanf("%d", &n);
    for (int row = 1; row <= n; row++)
    {
        for(int col = 1; col <= row; col++)
        { printf("* ");
        }
```

```
            printf("\n");
        }
}
```

... output :

```
Enter the value of n : 5
*
* *
* * *
* * * *
* * * * *
```

08. Pattern type : 05 -

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

... code :

```
# include <stdio.h>
int main()
{
    int n;
    printf("Enter the value of n :");
    scanf("%d", &n);

    for(int row=1; row <= n; row++)
    {   for(int col = 1; col <= row; col ++)
        { printf("%d ", row);
        }
        printf("\n");
    }
}
```

... output :

```
Enter the value of n: 4
1
2 2
3 3 3
4 4 4 4
```

08. Pattern type : 06 -

```
1
0 0
1 1 1
```

... code:

```
#include <stdio.h>
int main()
{
    int n;
    printf("Enter the value of n: ");
    scanf("%d", &n);

    for(int row=1; row<=n; row++)
    {
        for(int col=1; col<=row; col++)
        {
            if(row%2==0)
            {
                printf("0 ");
            }
            else
            {
                printf("1 ");
            } printf("\n");
        }
    }
}
```

... output:

```
Enter the value of n: 4
1
0 0
1 1 1
0 0 0 0
```

10. Pattern type - 07 -

```
A
B B
C C C
```

... Code:

```
#include <stdio.h>
int main()
{
   int n;
   printf("Enter the value of n : ");
   scanf("%d", &n);

   for (int row=1; row<=n; row++)
   {
      for (int col=1; col<=row; col++)
      {
         printf("%c ", row+64);
      }
      printf("\n");
   }
}
```

... output:

```
Enter the value of n: 5
A
B B
C C C
D D D D
E E E E E
```

11. Pattern type : 08 -

```
#
# #
# # #
```

... code:

```
#include <stdio.h>
int main(){
   int n;
```

```
    printf(" Enter the value of n : ");
    scanf("%d ", &n);

    for(int row = 1; row <= n ; row++).
       {
        for(int col = 1; col <= row; col++)
         {
           printf("# ");
         }
         printf("\n");
       }
}
```

... Output:

Enter the value of n : 5

```
#
# #
# # #
# # # #
# # # # #
```

Conclusion:

This lab report focuses on understanding the concept of primality and iterative problem-solving using loops in C programming. These programs illustrate diverse applications, reinforcing the ability to structure and solve real-world problem systematically. These foundational skills are critical for advanced computational challange.

References:

- C Standard library documentation.
- github.com
- freecodecamp.org
- stackoverflow.com