**Programs:**

**while loop**

1. WAP to display first n natural numbers using while loop(Natural numbers starts from 1)

#include<stdio.h>

int main()

{

int i=1,n;

printf("\n Enter n:");

scanf("%d",&n);

while(i<=n)

{

printf("\n%d",i);

i++;

}

return 0;

}

1. WAP to display sum of all numbers from 1 to n

#include<stdio.h>

int main()

{

int i=1,n,sum=0;

printf("\n Enter n:");

scanf("%d",&n);

while(i<=n)

{

sum=sum+i;

i++;

}

printf("\n Sum is:%d",sum);

return 0;

}

1. WAP to count all even numbers from m to n

#include<stdio.h>

int main()

{

int m,n,count=0;

printf("\n Enter m and n:");

scanf("%d %d",&m,&n);

while(m<=n)

{

if(m%2==0)

{

count++;

}

m++;

}

printf("\n Count of even numbers is:%d",count);

return 0;

}

1. WAP to display factorial of a number

#include<stdio.h>

int main()

{

long long int fact=1;

int n,i=1;

printf("\n Enter number:");

scanf("%d",&n);

if(n==0)

{

printf("\n Factorial is:1");

}

else

{

while(i<=n)

{

fact=fact\*i;

i++;

}

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

}

return 0;

}

1. WAP to count total number of digits in a number

#include<stdio.h>

int main()

{

int n,count=0;

printf("\n Enter number:");

scanf("%d",&n);

while(n!=0)

{

count++;

n=n/10;

}

printf("\n Total number of digits are:%d",count);

return 0;

}

1. WAP to find the sum of all digits of a number

#include<stdio.h>

int main()

{

int n,digit,sum=0;

printf("\n Enter number:");

scanf("%d",&n);

while(n!=0)

{

digit=n%10;

sum=sum+digit;

n=n/10;

}

printf("\n Sum of the digits is:%d",sum);

return 0;

}

1. WAP to display the reverse of a number

#include <stdio.h>

int main()

{

int n, reversedNumber = 0, remainder;

printf("Enter an integer: ");

scanf("%d", &n);

while(n != 0)

{

remainder = n%10;

reversedNumber = reversedNumber\*10 + remainder;

n = n/10;

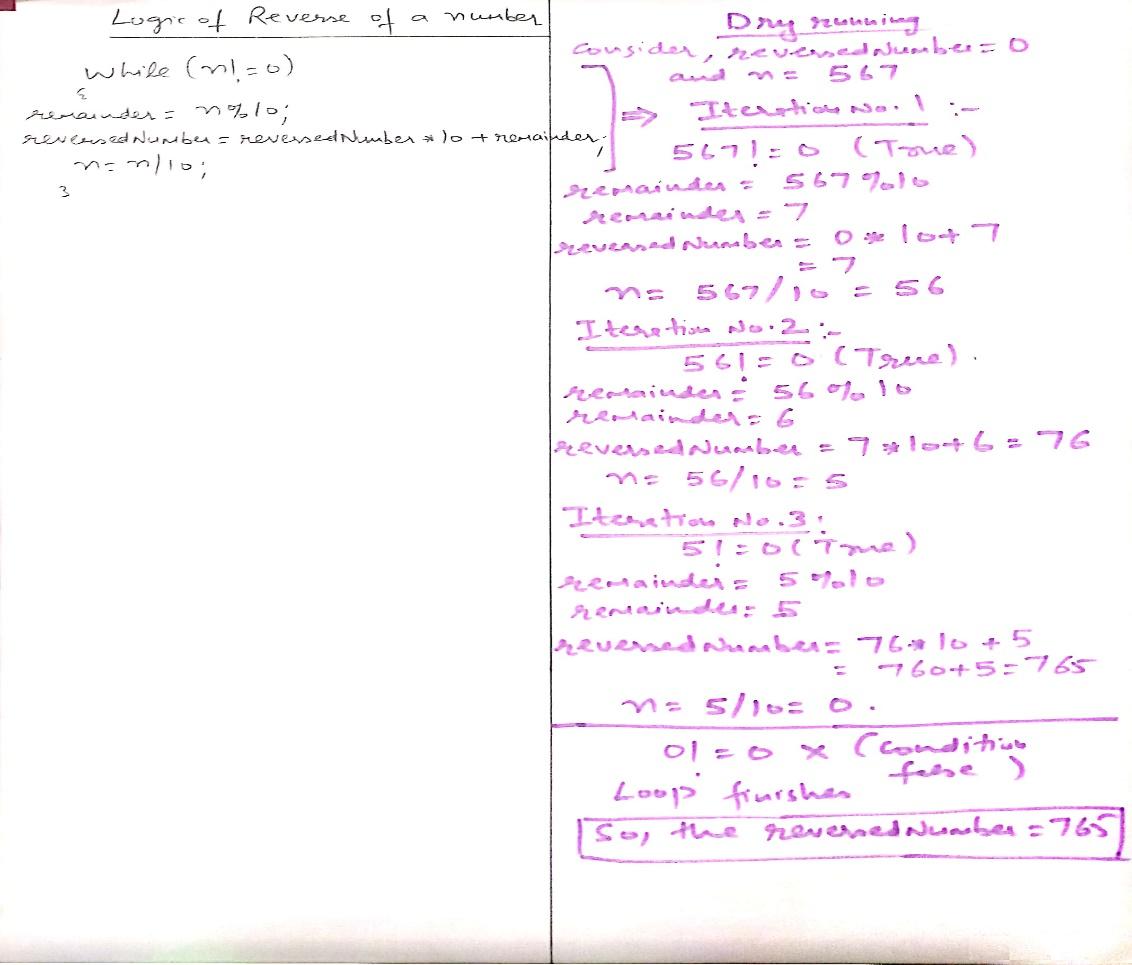
}

printf("Reversed Number = %d", reversedNumber);

return 0;

}

Dry Running of the logic (Explanation):



1. WAP to check whether the entered integer is palindrome or not

//Note: Palindrome number is that one whose reverse is equivalent to the original number

#include <stdio.h>

int main()

{

int n, reversedInteger = 0, remainder, originalInteger;

printf("Enter an integer: ");

scanf("%d", &n);

originalInteger = n;

// reversed integer is stored in variable

while( n!=0 )

{

remainder = n%10;

reversedInteger = reversedInteger\*10 + remainder;

n = n/10;

}

// palindrome if orignalInteger and reversedInteger are equal

if (originalInteger == reversedInteger)

printf("%d is a palindrome.", originalInteger);

else

printf("%d is not a palindrome.", originalInteger);

return 0;

}

1. WAP to check whether the entered number is Armstrong number or not

//Note: Armstrong number is that number which is equivalent to the sum of the powers of its digit, where power of each digit is dependent upon the count of the digits, e.g: 371(Here count of digits is: 3, so power is 3, now 3\*3\*3+7\*7\*7+1\*1\*1=27+343+1=371, another example:1634 (Here count of digits is:4, so power is 4, now 1\*1\*1\*1+6\*6\*6\*6+3\*3\*3\*3+4\*4\*4\*4=1+1296+81+256=1634)

#include <stdio.h>

#include <math.h>

int main()

{

int number, originalNumber, remainder, result = 0, n = 0 ;

printf("Enter an integer: ");

scanf("%d", &number);

originalNumber = number;

while (originalNumber != 0)

{

originalNumber =originalNumber/10;

n++;

}

originalNumber = number;

while (originalNumber != 0)

{

remainder = originalNumber%10;

result =result+pow(remainder, n);

originalNumber=originalNumber/10;

}

if(result == number)

printf("%d is an Armstrong number.", number);

else

printf("%d is not an Armstrong number.", number);

return 0;

}

**for loop**

1. WAP to check whether the entered number is prime or composite

#include <stdio.h>

int main()

{

int n, i, status = 0;

printf("Enter a positive integer: ");

scanf("%d",&n);

for(i=2; i<=n/2; i++)

{

// condition for nonprime number

if(n%i==0)

{

status=1;

break;

}

}

if (status==0)

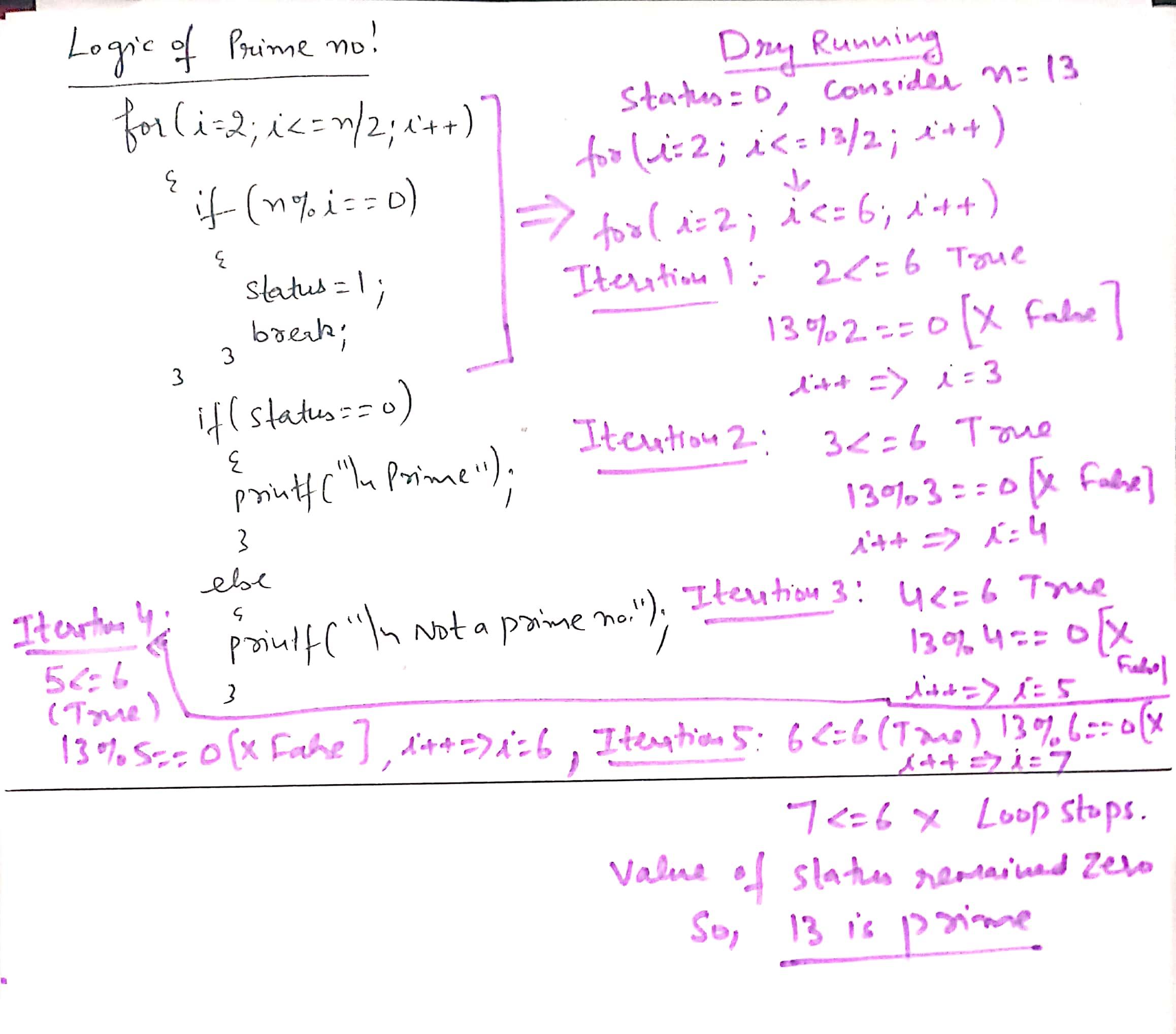
printf("%d is a prime number.",n);

else

printf("%d is not a prime number(or composite).",n);

return 0;

}

Dry running (Explanation of logic):

1. WAP to display n terms of Fibonacci series (e.g. if n=5, then fibonaaci terms are: 0 1 1 2 3 …..)

#include <stdio.h>

int main()

{

int i, n, t1 = 0, t2 = 1, nextTerm;

printf("Enter the number of terms: ");

scanf("%d", &n);

printf("Fibonacci Series: ");

for (i = 1; i <= n; i++)

{

printf("%d, ", t1);

nextTerm = t1 + t2;

t1 = t2;

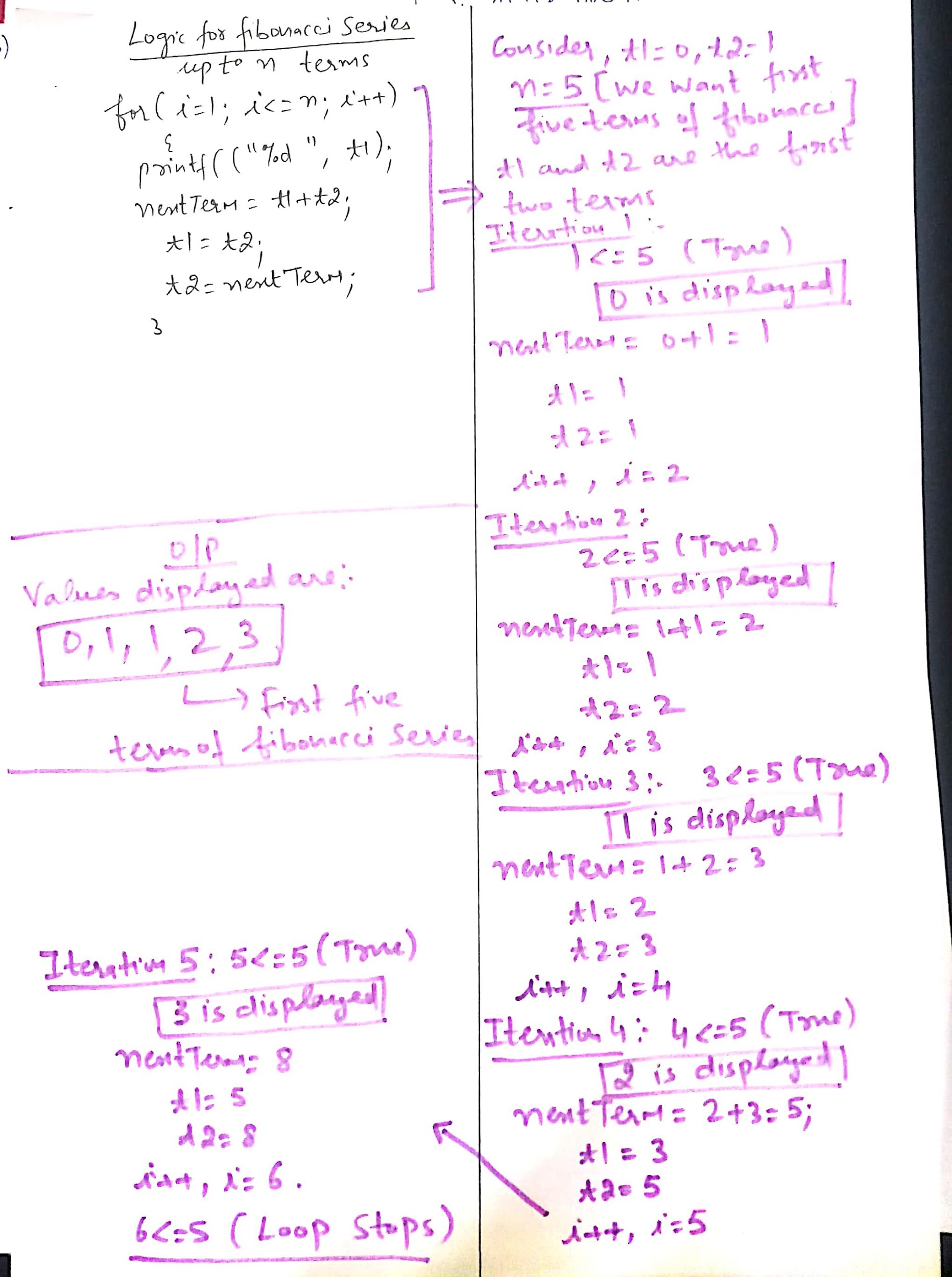
t2 = nextTerm;

}

return 0;

}

Dry running (Explanation of logic)



1. WAP to display multiplication table upto any limit for any integer: n

#include <stdio.h>

int main()

{

int n, i=1,limit=1;;

printf("Enter an integer: ");

scanf("%d",&n);

printf("\n Enter the limit:");

scanf("%d",&limit);

for(i=1;i<=limit;i++)

{

printf("%d X %d = %d \n", n, i, n\*i);

}

return 0;

}

**Practice questions:**

**Implement all the programs which are done using while loop with the help of for loop [i.e. from Program no. 1 to 9]**

**do while loop**

1. WAP to find the sum of all multiples of 5 from 1 to 100

#include <stdio.h>

int main()

{

int i=1,n,sum=0;

printf("\n Enter value of n:");

scanf("%d",&n);

do

{

if(i%5==0)

{

sum=sum+i;

}

i++;

}while(i<=n);

printf("\n Sum of all multiples of 5 is:%d",sum);

return 0;

}

1. WAP to count multiples of 7 from m to n

#include <stdio.h>

int main()

{

int m,n,count=0;

printf("\n Enter value of m and n:");

scanf("%d %d",&m,&n);

do

{

if(m%7==0)

{

count++;

}

m++;

}while(m<=n);

printf("\n Count of multiples of 7 is:%d",count);

return 0;

}

**Practice questions:**

**Implement all the programs which are done using while loop and for loop with the help of do-while loop [i.e. from Program no. 1 to 12]**

**Nested loops [Using loop inside loop]**

**We can display patterns with the help of nested loops/ or we can do all the operations where more one loops are required**

1. WAP to display following pattern using nested loops

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

#include <stdio.h>

int main()

{

int i,j;

for(i=1;i<=5;i++)

{

for(j=1;j<=i;j++)

{

printf("\*");

}

printf("\n");

}

return 0;

}

1. WAP to display following pattern using nested loops

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

#include <stdio.h>

int main()

{

int i,j;

for(i=1;i<=5;i++)

{

for(j=5;j>=i;j--)

{

printf("\*");

}

printf("\n");

}

return 0;

}

1. WAP to display following pattern using nested loops

1 1 1 1 1

2 2 2 2 2

3 3 3 3 3

4 4 4 4 4

5 5 5 5 5

#include <stdio.h>

int main()

{

int i,j;

for(i=1;i<=5;i++)

{

for(j=1;j<=5;j++)

{

printf("%d",i);

}

printf("\n");

}

return 0;

}

1. WAP to display following pattern using nested loops

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

#include <stdio.h>

int main()

{

int i,j;

for(i=1;i<=5;i++)

{

for(j=1;j<=i;j++)

{

printf("%d",j);

}

printf("\n");

}

return 0;

}

1. WAP to display following pattern with the help of nested loops

1 2 3 4 5

1 2 3 4

1 2 3

1 2

1

#include <stdio.h>

int main()

{

int i,j;

for(i=5;i>=1;i--)

{

for(j=1;j<=i;j++)

{

printf("%d",j);

}

printf("\n");

}

return 0;

}

1. WAP to display the following pattern with the help of nested loops

A

B B

C C C

D D D D

E E E E E

#include <stdio.h>

int main()

{

int i,j;

for(i=65;i<=69;i++)

{

for(j=65;j<=i;j++)

{

printf("%c",i);

}

printf("\n");

}

return 0;

}

1. WAP to count total number of prime numbers from 10 to 100

#include <stdio.h>

int main()

{

int i,j,status=0,count=0;

for(i=10;i<=100;i++)

{

for(j=2;j<=i/2;j++)

{

if(i%j==0)

{

status=1;

break;

}

}

if(status==0)

{

count++;

}

else

{

status=0;

}

}

printf("\n Total number of prime numbers from 10 to 100 are:%d",count);

return 0;

}

Practice questions to do:

WAP to display following patterns with the help of nested loops

1. A B C D E

A B C D

A B C

A B

A

1. 10 11 12 13

10 11 12

10 11

10

1. \* \* \* \* \*

# # # #

\* \* \*

# #

\*

d) a

a b

a b c

a b c d

* WAP to count all palindromes from 100 to 1000
* WAP to display all Armstrong numbers from 100 to 500

**Jump statements program examples:**

**break statement**

#include <stdio.h>

int main()

{

int I;

for(i=1;i<=10;i++)

{

if(i==6)

break;

printf("%d ",i);

}

return 0;

}

Output: 1 2 3 4 5

//Note after displaying 5, the value of i becomes 6, and when the condition is true break statement works which will terminate the loop and hence the loop ends

**continue statement**

#include <stdio.h>

int main()

{

int I;

for(i=1;i<=10;i++)

{

if(i%2==0)

continue;

printf("%d ",i);

}

return 0;

}

Output: 1 3 5 7 9

// Note this logic is displaying all odd numbers from 1 to 10, as when the value of i becomes even the continue statement works, which will skip the next set of statements and control is transferred to the next iteration of the loop, hence the even numbers are not printed, only odd numbers are printed.

**goto statement**

#include<stdio.h>

int main()

{

int x;

y:printf("\n Enter number:");

scanf("%d",&x);

if(x>100)

{

printf("\n Input is not ok");

goto y;

}

else

{

goto z;

}

z:printf("\n Number is less than or equal to 100");

return 0;

}

Note: This program will check the input, if the input is greater than 100, then control is transferred to input section again, as input should be less than or equal to 100, here y is the label for goto

**return statement**: It is used to return a value from a function, Even in main() function it is used to return a value, and normally it is placed at the end part of function body, because after return statement, no other statement will execute, as function will return the value. This statement can be used in user defined function, which can return the value obtained after doing some calculation in the function body.

#include<stdio.h>

int main()

{

printf("\n Checking return statement");

return 0;

printf("\n Statement after return");

}

Output: Checking return statement

Note: Statement after return 0 will not work as main() has returned the value and after that no statement will execute