**Algorithm Development and Programming Fundamentals**

**Term-work**

**Roll No :- MA028**

**Name : Nikhil R. Lathiya**

**1.** The Collatz function is defined for a positive integer n as follows.

f(n) = 3n+1 if n is odd,

n/2 if n is even

We consider the repeated application of the Collatz function starting with a

given integer n, as follows:

f(n), f(f(n)), f(f(f(n))), ...

It is conjectured that no matter which positive integer n you start from, this

sequence eventually will have 1 in it.

e.g. If n=7, the sequence is

1) f(7) = 22

2) f(f(7)) = f(22) = 11

3) f(11) = 34

4) f(34) = 17

5) f(17) = 52

6) f(52) = 26

7) f(26) = 13

8) f(13) = 40

9) f(40) = 20

10) f(20) = 10

11) f(10) = 5

12) f(5) = 16

13) f(16) = 8

14) f(8) = 4

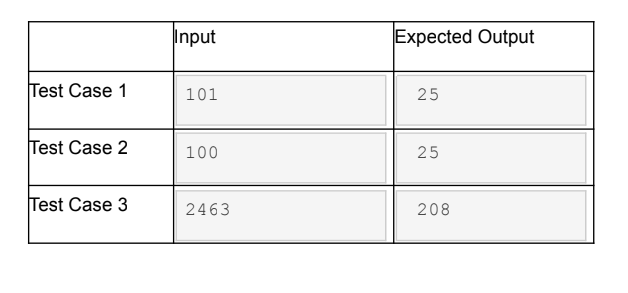
15) f(4) = 2

16) f(2) = 1

Thus if you start from n=7, you need to apply f 16 times in order to first get 1.

In this question, you will be given a positive number <= 32,000. You have to

output how many times f has to be applied repeatedly in order to first reach 1.



**Code:-**

#include<stdio.h>

int f(int n)

{

int l = 0;

while(n != 1)

{

if(n%2 == 0)

{

n = n / 2;

}else{

n = 3 \* n + 1;

}

l++;

}

return l;

}

int main()

{

int a,b;

scanf("%d",&a);

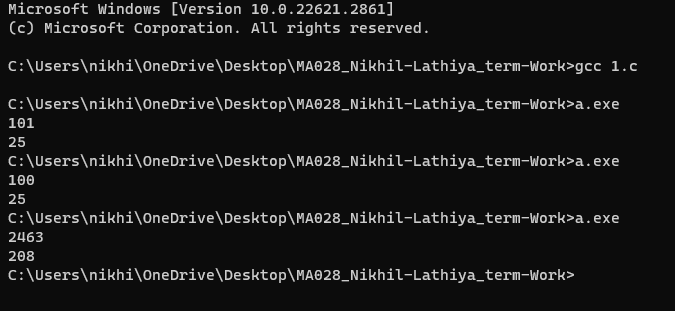
b= f(a);

printf("%d",b);

return 0;

}

**Output**



**2.** Write a recursive program that inputs a line of characters from the user. The

line may contain blanks. It outputs the line with the characters reversed. The

input ends with EOF (end of file).

NOTE: You have to use recursion to solve this, and are NOT allowed to use

array to store the input!!

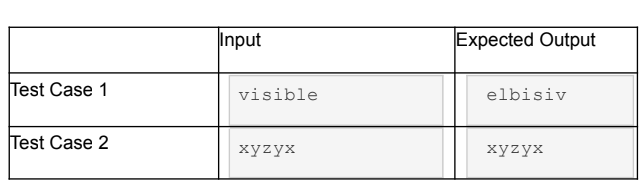
Example:

INPUT

This is easy

OUTPUT

ysae si sihT



**Code**

#include <stdio.h>

void fun()

{

char a;

scanf("%c", &a);

if (a == '\n')

{

return;

}

fun();

printf("%c", a);

}

int main()

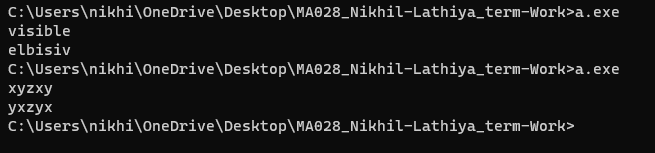
{

fun();

return 0;

}

**Output**



`

**3.** We say that a string 's' is an anagram of another string 't' if the letters in 's' can

be rearranged to form 't'.

For example, "butterfly" is an anagram of "flutterby", since a rearrangement of

the first word results in the second.

We say that a position 'i' in 's' and 't' match, if 's' is an anagram of 't', and

s[i]==t[i].

In this question, you will be given two words, 's' and 't'. You have to output the

number of matching positions if s is an anagram of t, and -1 if s is not an

anagram of t.

Input

The input consists of two lines. The first line contains the first string, with length

<= 100 characters. The second line contains the second string, with length

<=100 characters.

**Output**

If the first string is an anagram of the second string, then output the number of

matching positions. Otherwise, print -1.

Sample Input 1

butterfly

flutterby

Sample Output 1

2

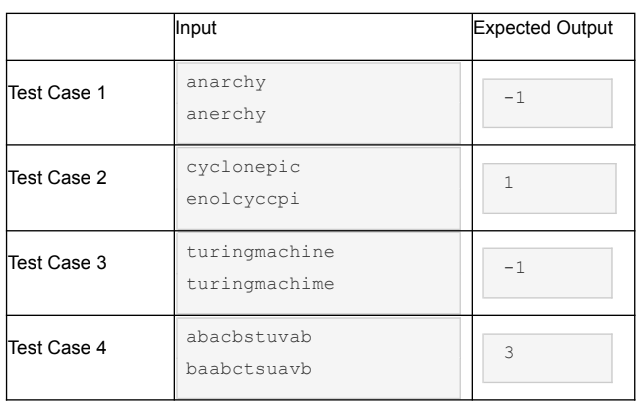
Sample Input 2

home

come

Sample Output 2

-1



**Code**

#include <stdio.h>

#include <string.h>

int main()

{

char s[100], t[100];

int i, j, ls, lt, match = 0, cnt = 0;

scanf("%s", s);

scanf("%s", t);

ls = strlen(s);

lt = strlen(t);

if (ls != lt)

{

printf("-1");

}

else

{

for (i = 0; i < ls; i++)

{

for (j = 0; j < lt; j++)

{

if (s[i] == t[j])

{

t[j] = '\0';

if (i == j)

{

match++;

}

cnt++;

break;

}

}

}

if (cnt == ls)

{

printf("%d", match);

}

else

{

printf("-1");

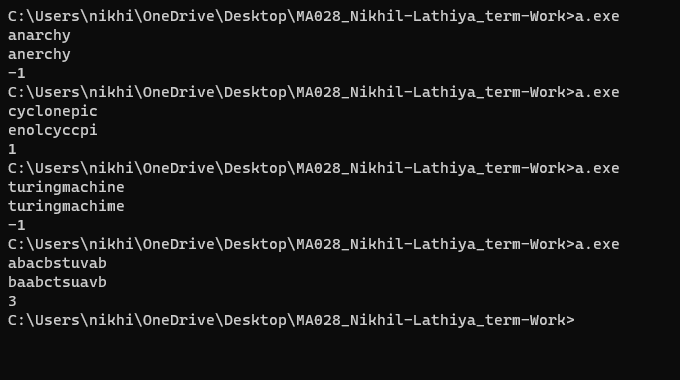
}

}

return 0;

}

**Output**



**4**. In a string, a "run" is a substring consisting of consecutive occurrences of the

same character. For example, the string "mississippi" contains the following

runs - "ss", "ss" and "pp".

In this question, given a string, you have to output the length of the longest run

in the string.

Input

A string, having length at most 100. The string is guaranteed to have

at least one run.

Output

The length of the longest run in the string.

Sample Input

abbaaacccc

Sample Output

4



**Code**

#include<stdio.h>

int main()

{

char str[100];

int cl=1,ml=1,i;

scanf("%s",str);

for(i=1;str[i] != '\0';++i)

{

if(str[i] == str[i-1])

{

cl++;

}else{

cl = 1;

}

if(cl > ml)

{

ml = cl;

}

}

printf("%d",ml);

return 0;

}

**Output**

A computer screen with white text

Description automatically generated

**5**. In this question, you are given two positive integers M and N, where M

< N. You may assume that N is less than or equal to 100.

The orbit of M with respect to N is defined to be the sequence

M, (2\*M) mod N, (2^2\*M) mod N, ...

There are at most N elements in the sequence, but for some M, the number of

elements in this sequence may be fewer.

You have to output the maximum number of distinct integers in the

orbit of M.

For example, if M=5 and N=8, then the orbit of 5 with respect to 8 is

5, 2\*5 mod 8, 4\*5 mod 8, 8\*5 mod 8

which is equal to

5, 2, 4, 0.

Hence the number of distinct integers in the orbit of 5 is 4.



**Code**

#include <stdio.h>

#include <math.h>

int count\_distinct(int \*arr, int n)

{

int i, j, cnt = 0;

for (i = 0; i < n; i++)

{

for (j = 0; j < i; j++)

{

if (arr[i] == arr[j])

{

break;

}

}

if (i == j)

{

cnt++;

}

}

return cnt;

}

int main()

{

int M, N;

scanf("%d %d", &M, &N);

int arr[N];

for (int i = 0; i < N; i++)

{

arr[i] = (M \* (int)pow(2, i)) % N;

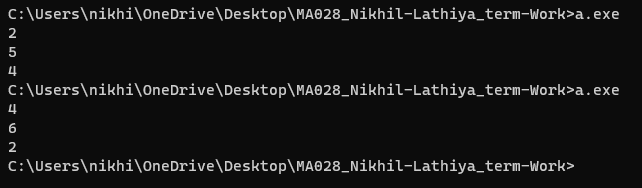
}

printf("%d", count\_distinct(arr, N));

return 0;

}

**Output**



**6**. You will be given an NxN matrix. You have to determine whether the

matrix is a triangular matrix.

The diagonal of the matrix M of size NxN is the set of entries M(0,0),

M(1,1), M(2,2), ..., M(N,N).

A matrix is upper triangular if every entry below the diagonal is 0. For

example,

1 1 1

0 0 1

0 0 2

is an upper triangular matrix. (The diagonal itself, and the entries above

and below the diagonals can be zeroes or non-zero integers.)

A matrix is lower triangular if every entry above the diagonal is 0. For

example,

2 0 0

3 1 0

4 2 2

is a lower triangular matrix.

A matrix is triangular if it is either upper triangular or lower triangular or

both.

You may not use arrays for this program.

Input

First, you will be given N, which is the size of the matrix.

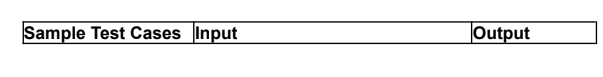
Then you will be given N rows of integers, where each row consists of

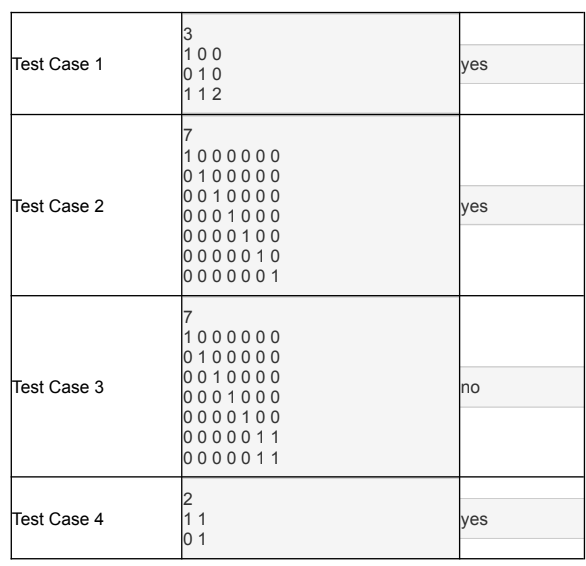
N integers separated by spaces.

Output

If the input matrix is triangular, then print yes. Otherwise, print

no.





**Code**

#include <stdio.h>

int main() {

int N;

printf("Enter the size of the matrix (N): ");

scanf("%d", &N);

int isUpperTriangular = 1

int isLowerTriangular = 1;

for (int i = 0; i < N; ++i) {

for (int j = 0; j < N; ++j) {

int num;

scanf("%d", &num);

// Check upper triangular condition

if (i > j && num != 0) {

isUpperTriangular = 0;

}

// Check lower triangular condition

if (i < j && num != 0) {

isLowerTriangular = 0;

}

}

}

if (isUpperTriangular || isLowerTriangular) {

printf("Yes\n");

} else {

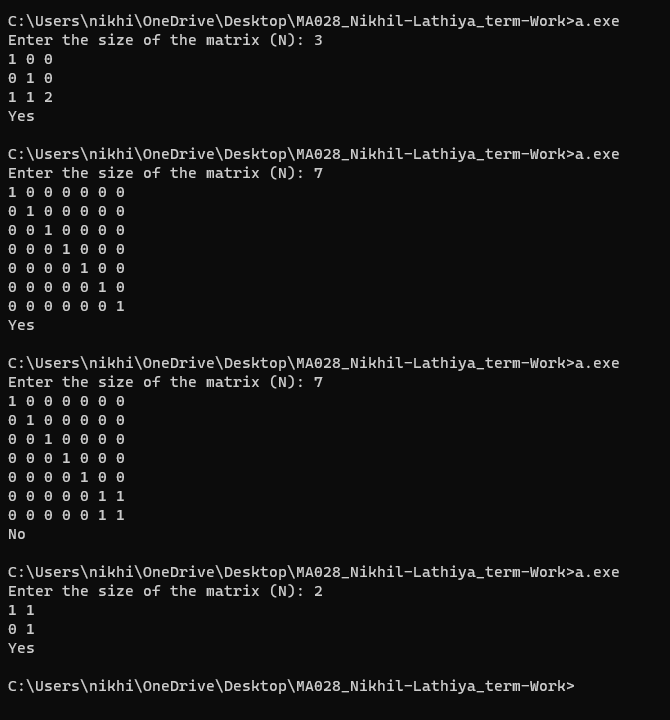
printf("No\n");

}

return 0;

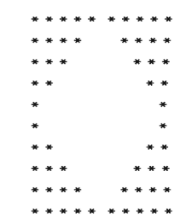
}

**Output**



**7**. Write Program to generate following pattern for input size N.

For N = 5 output is:



**Code**

#include <stdio.h>

int main()

{

int i, j, k;

for (i = 0; i < 5; i++)

{

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

{

printf("\* ");

}

for (k = 0; k < i; k++)

{

printf(" ");

}

for (j = 0; j < i; j++)

{

printf(" ");

}

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

{

printf("\* ");

}

printf("\n");

}

for (i = 0; i < 5; i++)

{

for (j = 0; j <= i; j++)

{

printf("\* ");

}

for (k = 4; k > i; k--)

{

printf(" ");

}

for (j = 4; j > i; j--)

{

printf(" ");

}

for (k = 0; k <= i; k++)

{

printf("\* ");

}

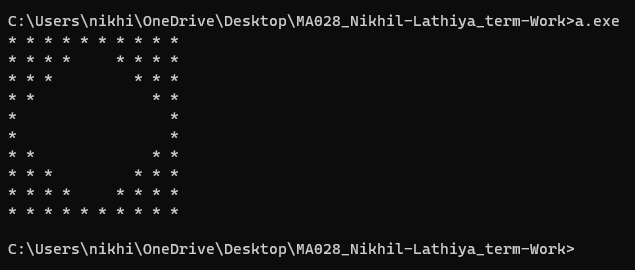
printf("\n");

}

return 0;

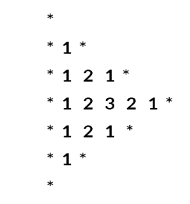
}

**Output**



**8**. Write Program to generate following pattern for input size N.

For N=3 output is:



**Code**

#include<stdio.h>

int main()

{

int i,j,k,d=2;

printf("\*\n");

for(i=1;i<=3;i++)

{

if(i==0)

{

printf("\*");

}else{

printf("\*");

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

{

printf("%d",j);

}

for(k=j-2;k>=1;k--)

{

printf("%d",k);

}

printf("\*");}

printf("\n");

}

for(i=i-2;i>=0;i--)

{

if(i==0)

{

printf("\*");

}else{

printf("\*");

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

{

printf("%d",j);

}

for(k=j-2;k>=1;k--)

{

printf("%d",k);

}

printf("\*");

}

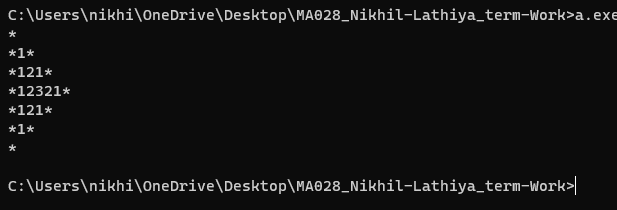
printf("\n");

}

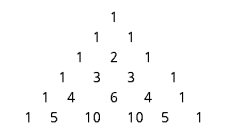
return 0;

}

**Output**



**9**. Write Program to generate the following pattern for input size N(rows).



**Code**

#include <stdio.h>

int main()

{

int rows;

printf("enter one positive number : ");

scanf("%d",&rows);

for (int i = 1; i <= rows; i++) {

for (int j = 0; j < rows - i; j++) {

printf(" ");

}

int C = 1;

for (int k = 1; k <= i; k++) {

printf("%d ", C);

C = C \* (i - k) / k;

}

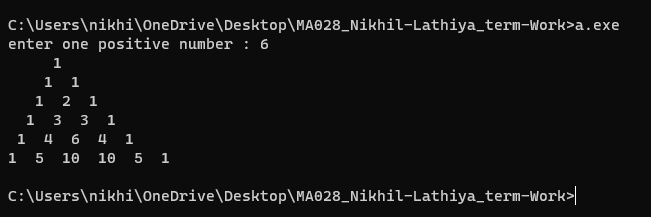
printf("\n");

}

return 0;

}

**Output**



**10**. Write a C program to find G.C.D of a Number - N using Recursion.

**Code**

#include<stdio.h>

int findgcd(int a,int b)

{

if(b == 0)

{

return a;

}else{

return findgcd(b, a % b);

}

}

int main()

{

int n1,n2;

printf("Enter first number: ");

scanf("%d", &n1);

printf("Enter second number: ");

scanf("%d", &n2);

if (n1 < 0 || n2 < 0) {

printf("Please enter non-negative numbers.\n");

return 1;

}

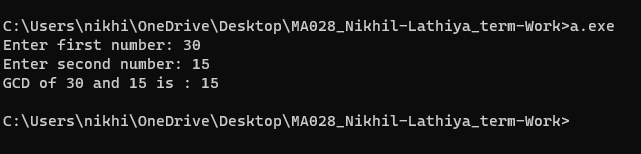
int gcd = findgcd(n1, n2);

printf("GCD of %d and %d is : %d\n",n1,n2,gcd);

return 0;

}

**Output**



**11**. Write a C program to print Fibonacci Series up to N terms using

Recursion.

**Code**

#include<stdio.h>

int fibo(int n)

{

if(n <= 1)

{

return n ;

}

else{

return fibo(n-1) + fibo(n-2);

}

}

void printfibo(int n)

{

printf("fibonacci series up to %d terms : ",n);

for(int i = 0;i <n;i++)

{

printf("%d ",fibo(i));

}

printf("\n");

}

int main()

{

int n;

printf("enter number of fibonacci series : ");

scanf("%d",&n);

if(n < 0)

{

printf("please enter non nagetive number.\n");

return 1;

}

printfibo(n);

return 0;

}

**Output**

