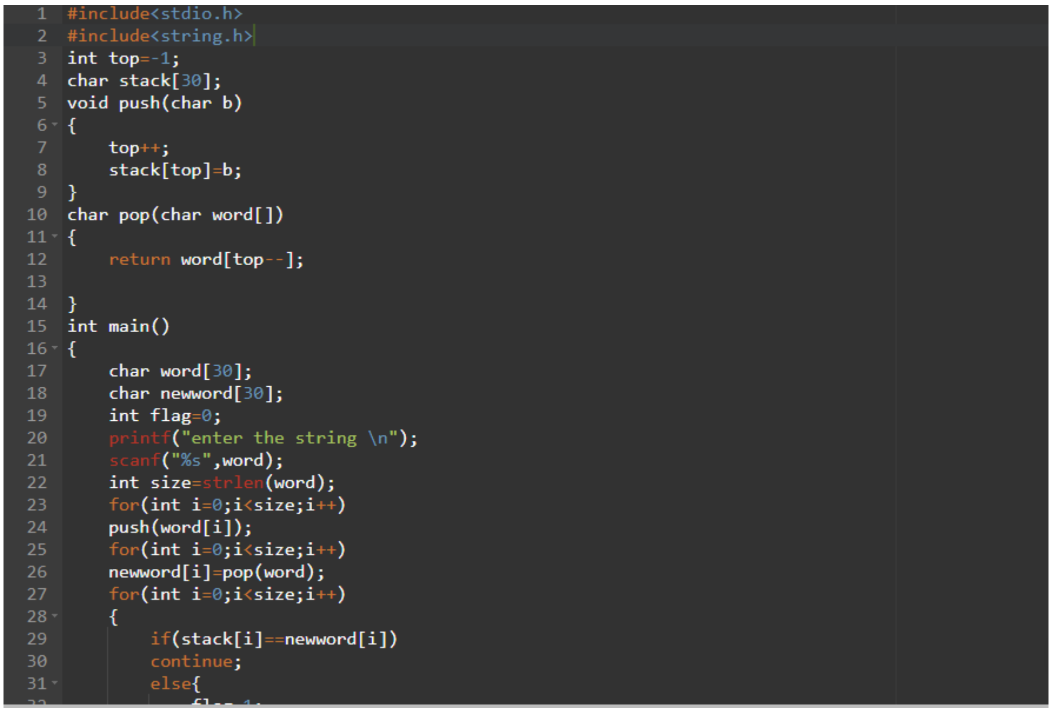
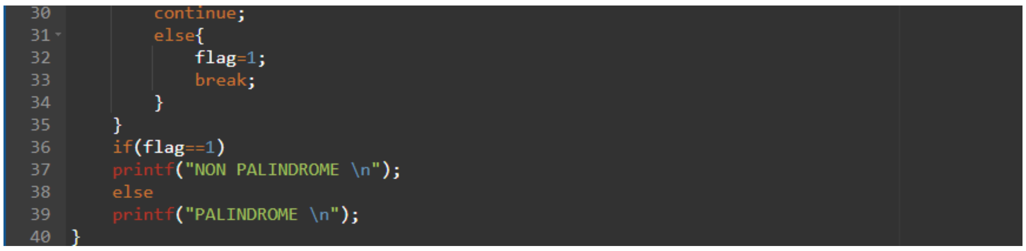
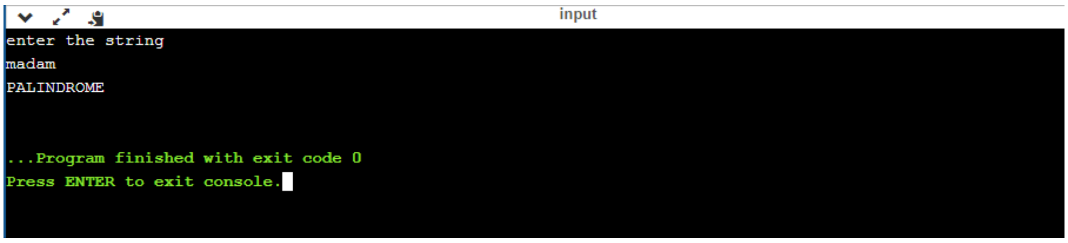


```c
#include<stdio.h>
#include<string.h>
int top=-1;
char stack[30];
void push(char b)
{
    top++;
    stack[top]=b;
}
char pop(char word[])
{
    return word[top--];

}
int main()
{
    char word[30];
    char newword[30];
    int flag=0;
    printf("enter the string \n");
    scanf("%s",word);
    int size=strlen(word);
    for(int i=0;i<size;i++)
    push(word[i]);
    for(int i=0;i<size;i++)
    newword[i]=pop(word);
    for(int i=0;i<size;i++)
    {
        if(stack[i]==newword[i])
        continue;
        else{
```

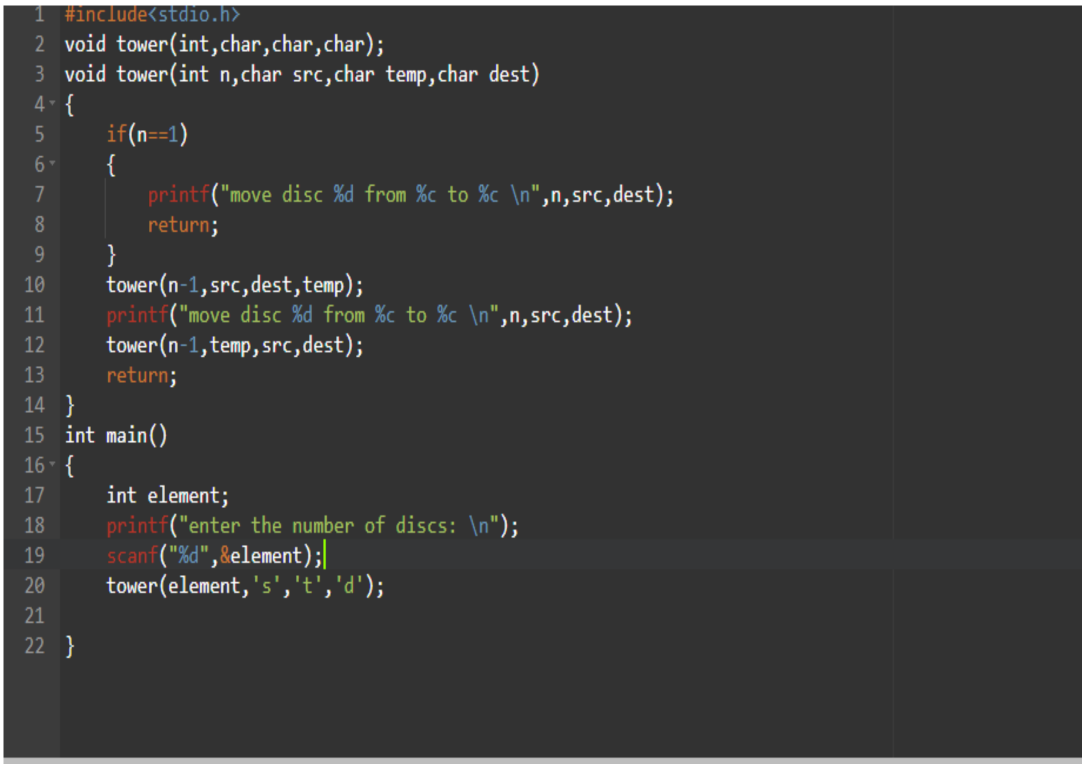


```c
        continue;
        else{
            flag=1;
            break;
        }
    }
    if(flag==1)
    printf("NON PALINDROME \n");
    else
    printf("PALINDROME \n");
}
```

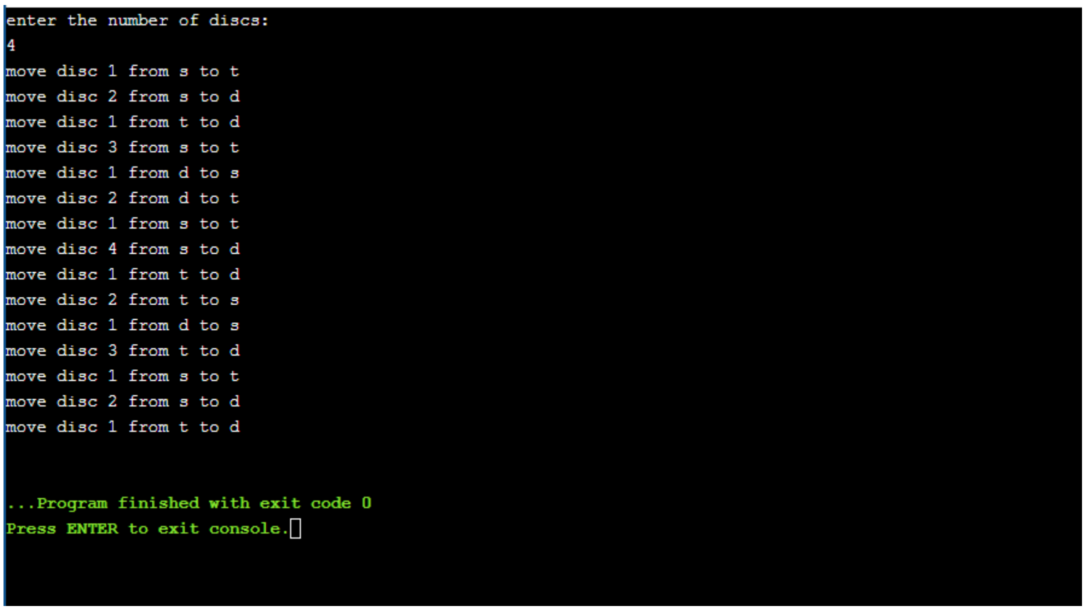


```
enter the string
madam
PALINDROME


...Program finished with exit code 0
Press ENTER to exit console.
```

Tower of hannoi:



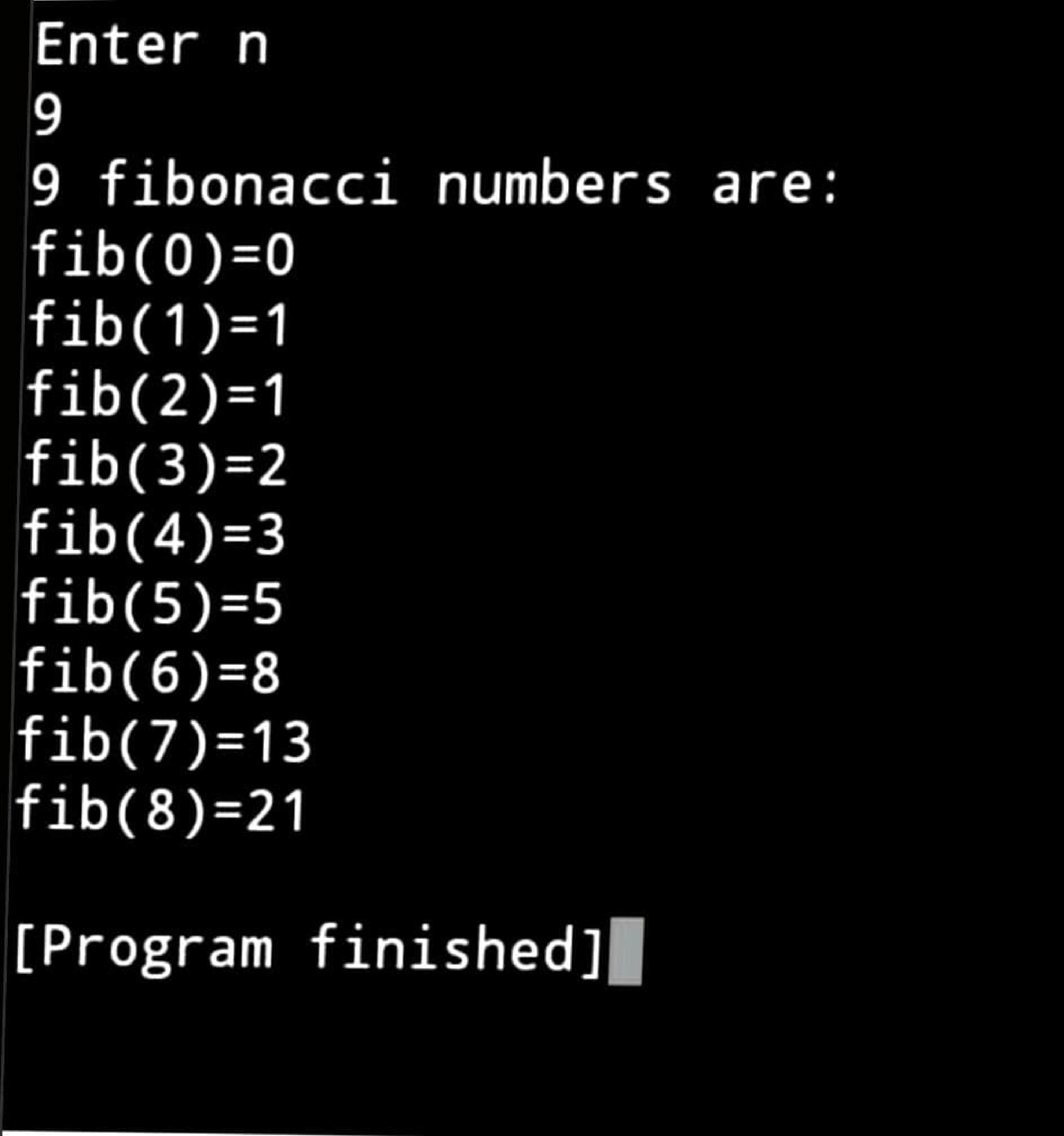
```c
#include<stdio.h>
void tower(int,char,char,char);
void tower(int n,char src,char temp,char dest)
{
    if(n==1)
    {
        printf("move disc %d from %c to %c \n",n,src,dest);
        return;
    }
    tower(n-1,src,dest,temp);
    printf("move disc %d from %c to %c \n",n,src,dest);
    tower(n-1,temp,src,dest);
    return;
}
int main()
{
    int element;
    printf("enter the number of discs: \n");
    scanf("%d",&element);
    tower(element,'s','t','d');

}
```



```
enter the number of discs:
4
move disc 1 from s to t
move disc 2 from s to d
move disc 1 from t to d
move disc 3 from s to t
move disc 1 from d to s
move disc 2 from d to t
move disc 1 from s to t
move disc 4 from s to d
move disc 1 from t to d
move disc 2 from t to s
move disc 1 from d to s
move disc 3 from t to d
move disc 1 from s to t
move disc 2 from s to d
move disc 1 from t to d


...Program finished with exit code 0
Press ENTER to exit console.
```

Fibonaci series :

```
#include<stdio.h>
int fib (int n)
 {
  if (n ==0)
     return0;
  if (n==1)
     return1;
 return fib (n-1 + fib(n-2) ;
}

int main()
{
int i, n;
  printf ("Enter n\n");
  scanf ("%d", &n);
printf ("%d fibonacci numbers are:
       \n", n);
   for ( i=0; i<n; i++) {
     printf ("fib(%d) = %d\n", i, fib(i))
   }
 }
```



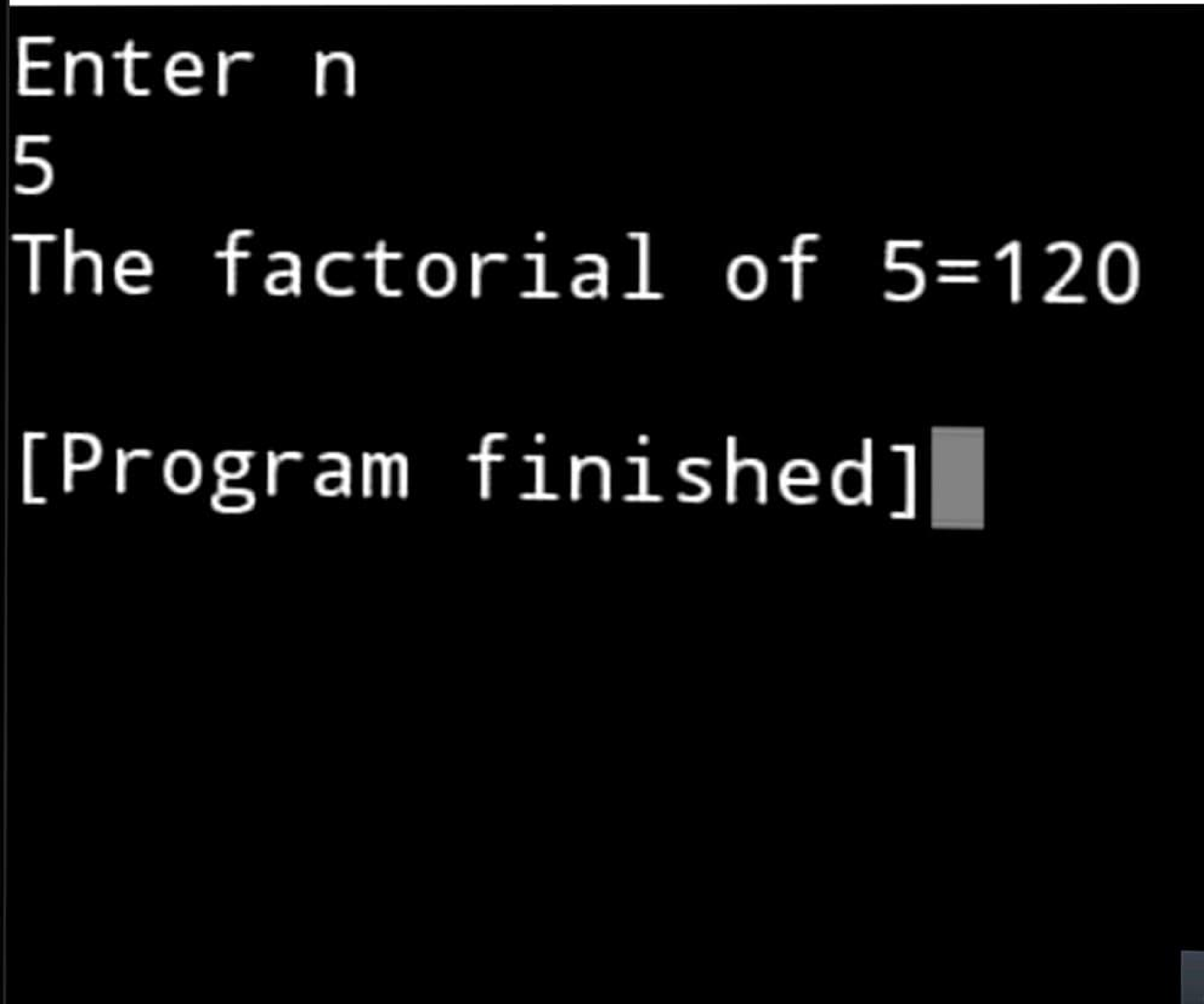
```
Enter n
9
9 fibonacci numbers are:
fib(0)=0
fib(1)=1
fib(2)=1
fib(3)=2
fib(4)=3
fib(5)=5
fib(6)=8
fib(7)=13
fib(8)=21

[Program finished]
```

Factorial of n numbers:

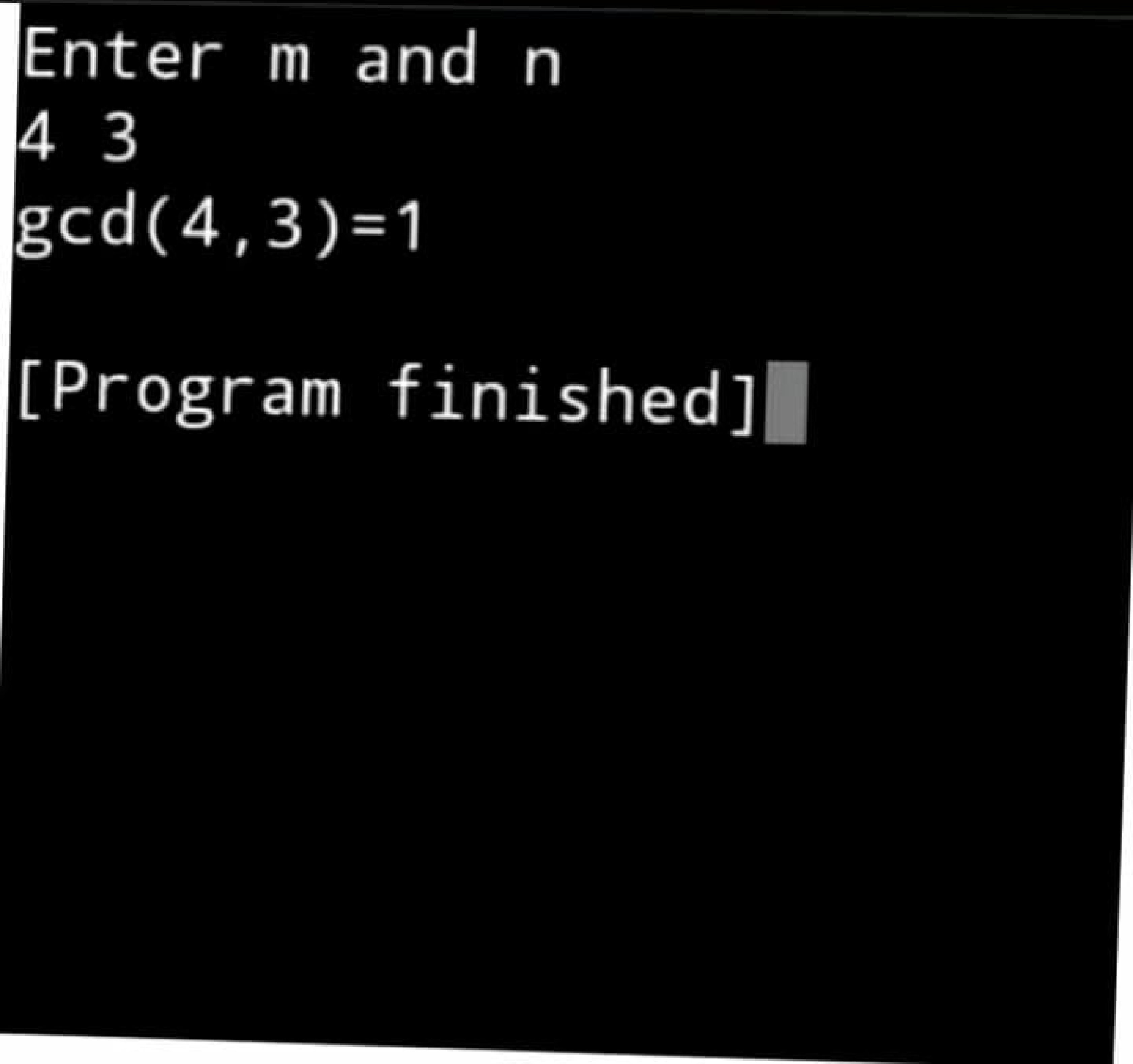
```
#include <stdio.h>
int fact (int n)
{
  if (n==0)
     return 1;
  return n * fact (n-1);
}
 int main()
{
  int n;
  printf ("enter the value of n \n");
  scanf ("%d", &n);
  printf ("The factorial of %d = %d\n",
           n, fact (n));
}
```

GCD of 2 nos

```
#include<stdio.h>
int gcd (int m, int n)
{
  if (n==0) return m;
  if (m<n) return gcd(n,m);
  return gcd (n, m%n);
}
 int main ()
{
  int m,n,res;
  printf ("enter m & n \n");
  scanf ("%d %d", &m, &n);
  res = gcd (m,n);
printf (" gcd (%d, %d) = %d\n", m,n,
                                res);
}
```



```
Enter n
5
The factorial of 5=120

[Program finished]
```

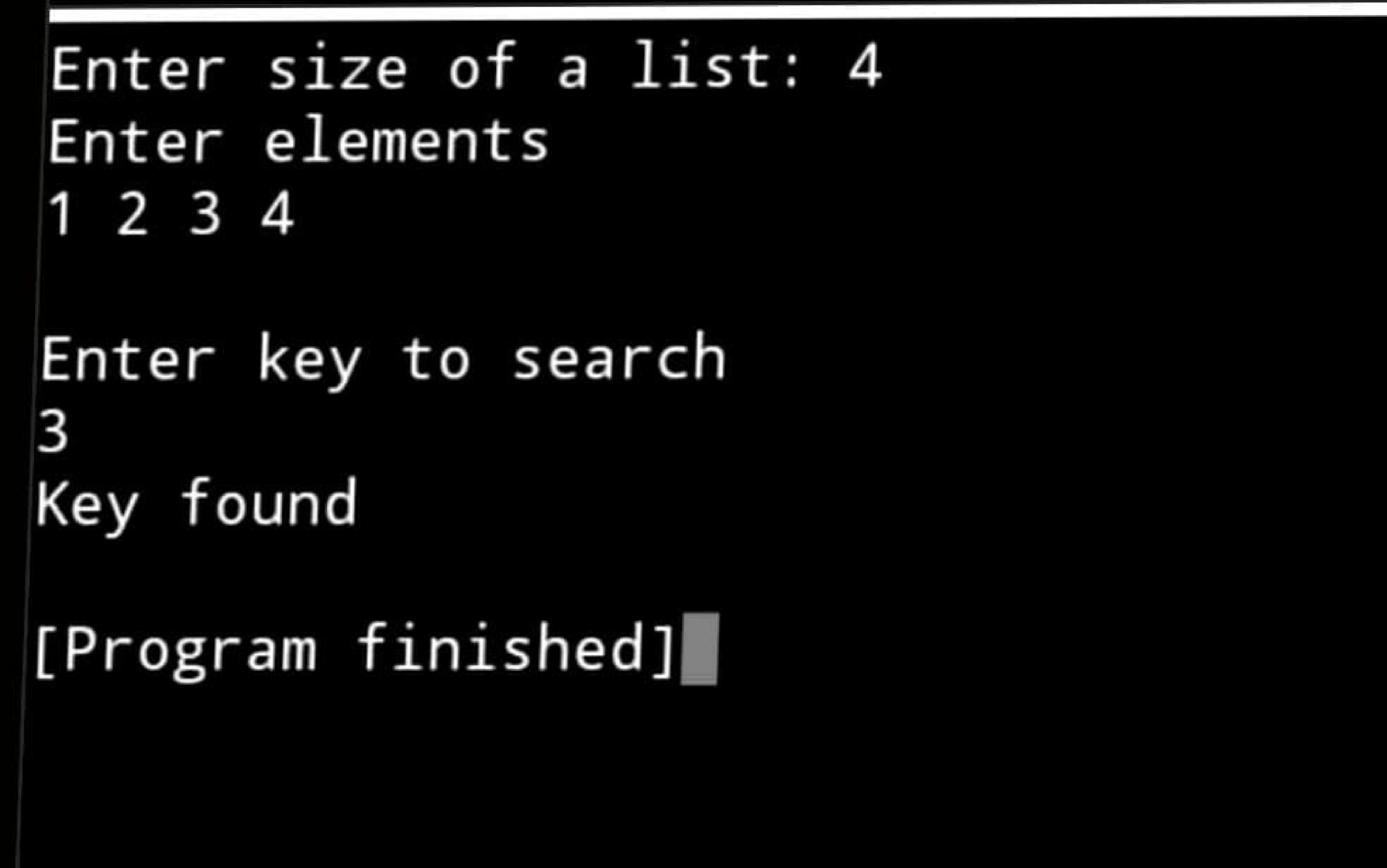


```
Enter m and n
4 3
gcd(4,3)=1

[Program finished]
```

Date: / /

Binary search using recursion:

```
#include <stdio.h>
void binary_search (int[], int int, int);
void bubblesort (int [], int);
 int main ()
 {
   int key, size, i;
   int list [25];
  printf ("enter size of a list :");
  scanf ("%d", & size);
  printf ("enter elements \n");
   for (i =0 ; i< size ; i++)
   {
    scanf ("%d", & list [i]);
   }
  bubble sort (list, size);
   printf ("\n");
   printf ("enter key to search \n");
  scanf ("%d", & key);
  binary search (list, 0, size, key);
}

void bubble sort (int list [ ], int size)
{
  int temp, i, j;
  for (i=0; i< size ; i++)
  {
     for ( j =1 ; j< size ; j++)
   {
   if (list [i] > list [j])
   {
       temp = list [ i ];
list[i] = list[j];
list[j] = temp;
}
}
}
}
void binary_search(int list[], int lo, int hi
                                       int key)
{
  int mid;
  if (lo > hi)
   {
 printf ("key not found \n");
  return;
 }
  mid = (lo + hi) / 2;
 if (list[mid] == key)
   {
    printf ("key found \n");
  }
   else if (list[mid > key)
   {
    binary_search(list, lo, mid-1, key)
   }
   else if (list[mid] < key)
   {
    binary_search(list, mid+1, hi, key);
   }
 }
```



```
Enter size of a list: 4
Enter elements
1 2 3 4


Enter key to search
3
Key found


[Program finished]
```