AMRITA SCHOOL OF COMPUTING

DESIGN AND ANALYSIS OF ALGORITHMS

(23CSE211)

Name: Y.V.S.Likesh

Roll No.: CH.SC.U4CSE24152

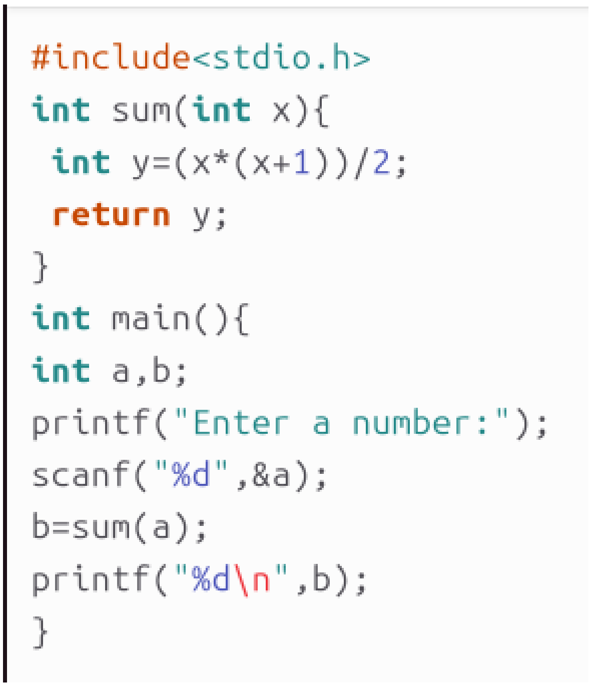
Class: BTech (CSE-B)

School: Amrita School of Computing,
Chennai Campus.

LAB-1

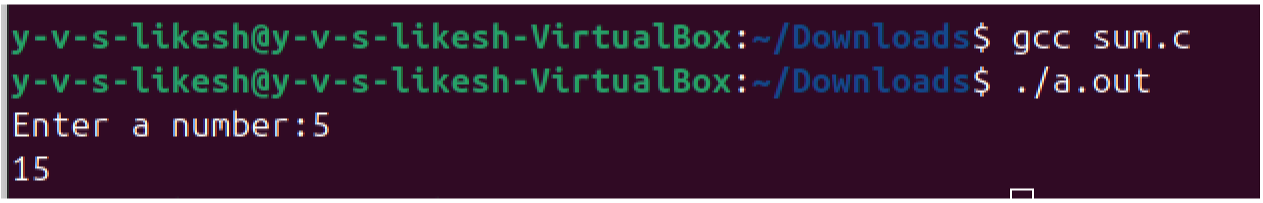
1) Write a program to find sum of n natural numbers (using user defined function)

Code:



```
#include<stdio.h>
int sum(int x){
 int y=(x*(x+1))/2;
 return y;
}
int main(){
int a,b;
printf("Enter a number:");
scanf("%d",&a);
b=sum(a);
printf("%d\n",b);
}
```

Output:



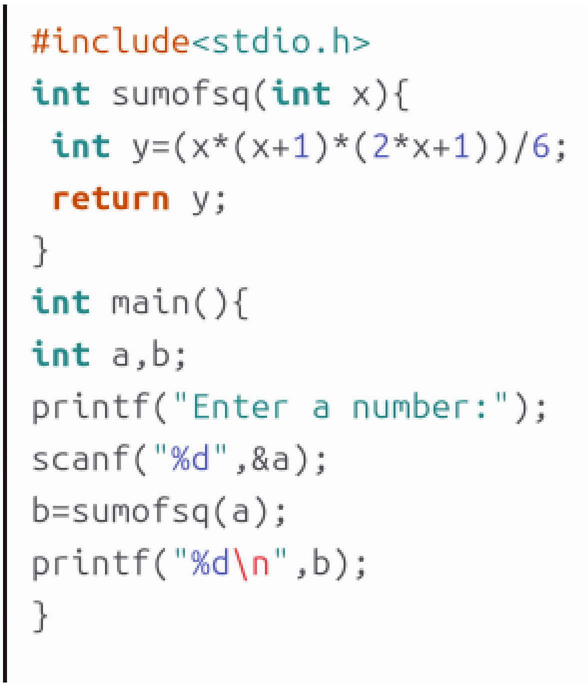
```
y-v-s-likesh@y-v-s-likesh-VirtualBox:~/Downloads$ gcc sum.c
y-v-s-likesh@y-v-s-likesh-VirtualBox:~/Downloads$ ./a.out
Enter a number:5
15
```

Space Complexity:

The space complexity of this program is O(1) - constant space. Because the program uses a fixed number of variables, independent of the input size, its space usage does not grow.

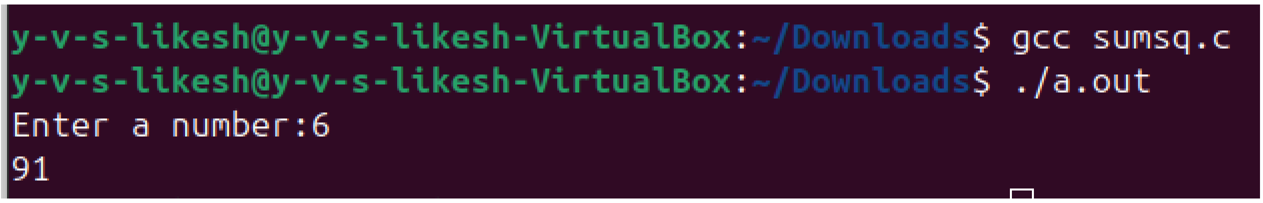
2) Write a program to find sum of squares of first n natural numbers

Code:



```
#include<stdio.h>
int sumofsq(int x){
 int y=(x*(x+1)*(2*x+1))/6;
 return y;
}
int main(){
int a,b;
printf("Enter a number:");
scanf("%d",&a);
b=sumofsq(a);
printf("%d\n",b);
}
```

Output:



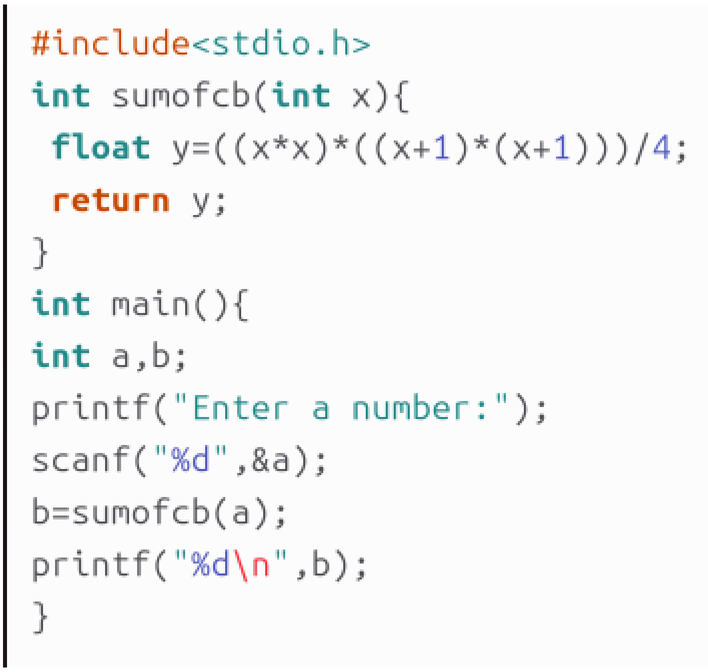
```
y-v-s-likesh@y-v-s-likesh-VirtualBox:~/Downloads$ gcc sumsq.c
y-v-s-likesh@y-v-s-likesh-VirtualBox:~/Downloads$ ./a.out
Enter a number:6
91
```

Space Complexity:

The space complexity of this program is O(1) - constant space. Because there is no recursion, no arrays, no dynamic memory allocation, the amount of memory used does not depend on n.

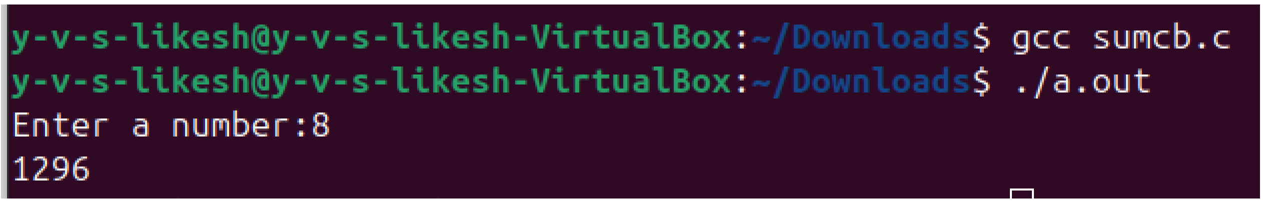
3) Write a program to find sum of cubes of first n natural numbers

Code:



```
#include<stdio.h>
int sumofcb(int x){
 float y=((x*x)*((x+1)*(x+1)))/4;
 return y;
}
int main(){
int a,b;
printf("Enter a number:");
scanf("%d",&a);
b=sumofcb(a);
printf("%d\n",b);
}
```

Output:



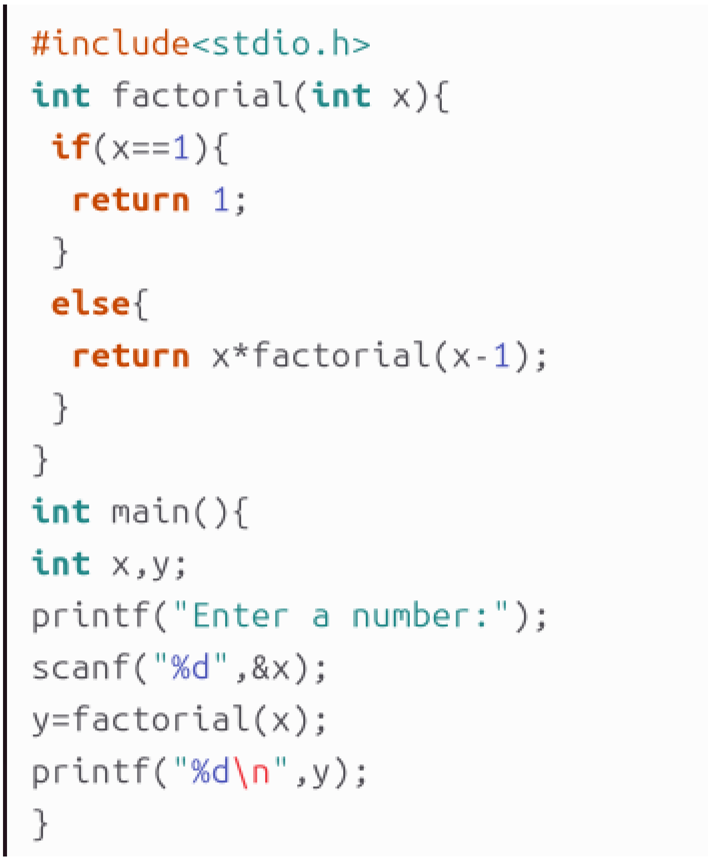
```
y-v-s-likesh@y-v-s-likesh-VirtualBox:~/Downloads$ gcc sumcb.c
y-v-s-likesh@y-v-s-likesh-VirtualBox:~/Downloads$ ./a.out
Enter a number:8
1296
```

Space Complexity:

The space complexity of this program is O(1) - constant space. Because there is no recursion, no arrays, no dynamic memory allocation, the amount of memory used does not depend on n.

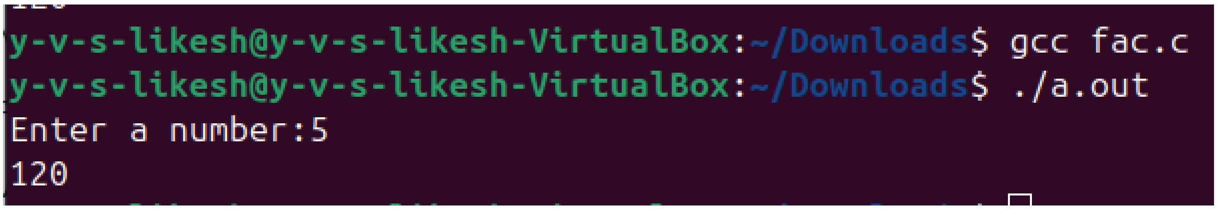
4) Write a program to find a factorial of given integer using recursion

Code:



```
#include<stdio.h>
int factorial(int x){
 if(x==1){
  return 1;
 }
 else{
  return x*factorial(x-1);
 }
}
int main(){
int x,y;
printf("Enter a number:");
scanf("%d",&x);
y=factorial(x);
printf("%d\n",y);
}
```

Output:



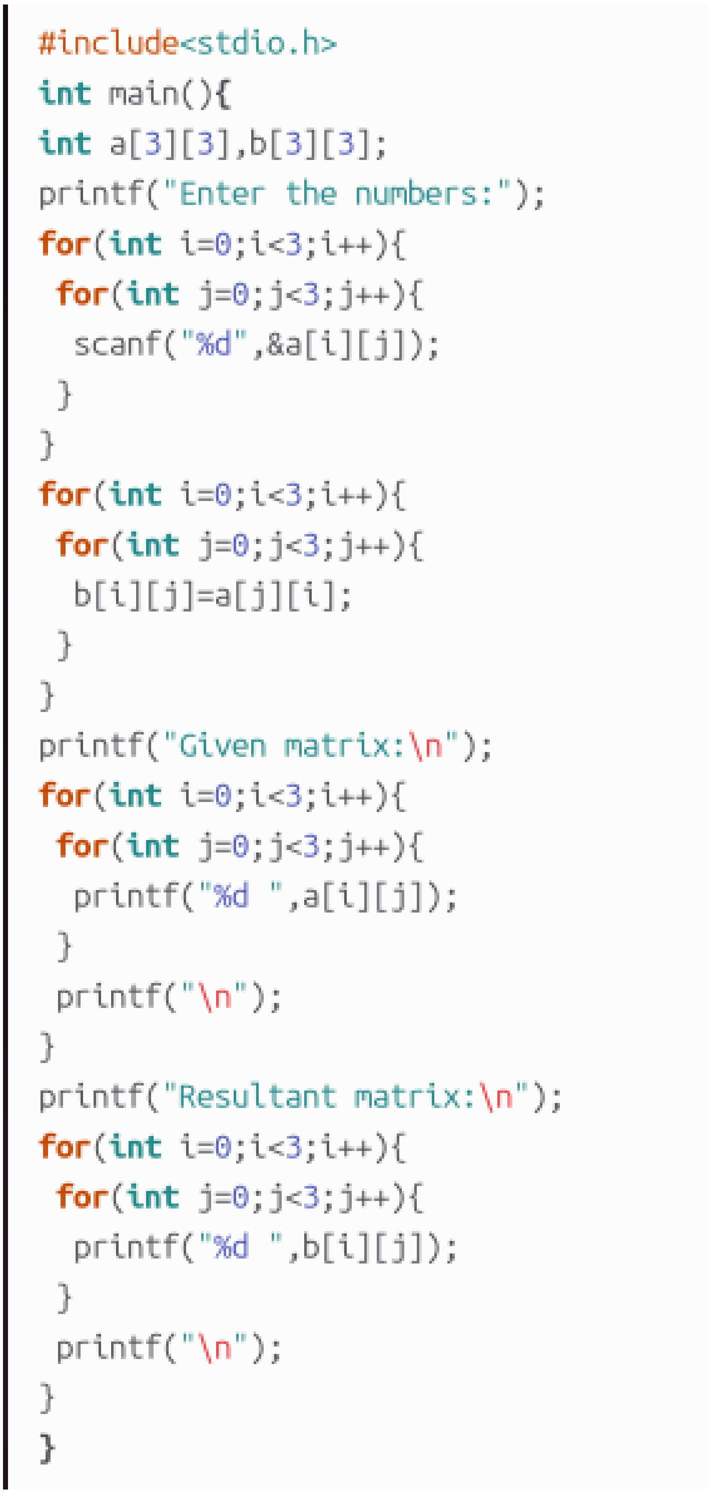
```
y-v-s-likesh@y-v-s-likesh-VirtualBox:~/Downloads$ gcc fac.c
y-v-s-likesh@y-v-s-likesh-VirtualBox:~/Downloads$ ./a.out
Enter a number:5
120
```

Space Complexity:

The space complexity of this problem is O(n). There are no arrays or dynamic memory allocation, but the recursive call stack makes the memory usage depend on n, resulting in O(n) space complexity.

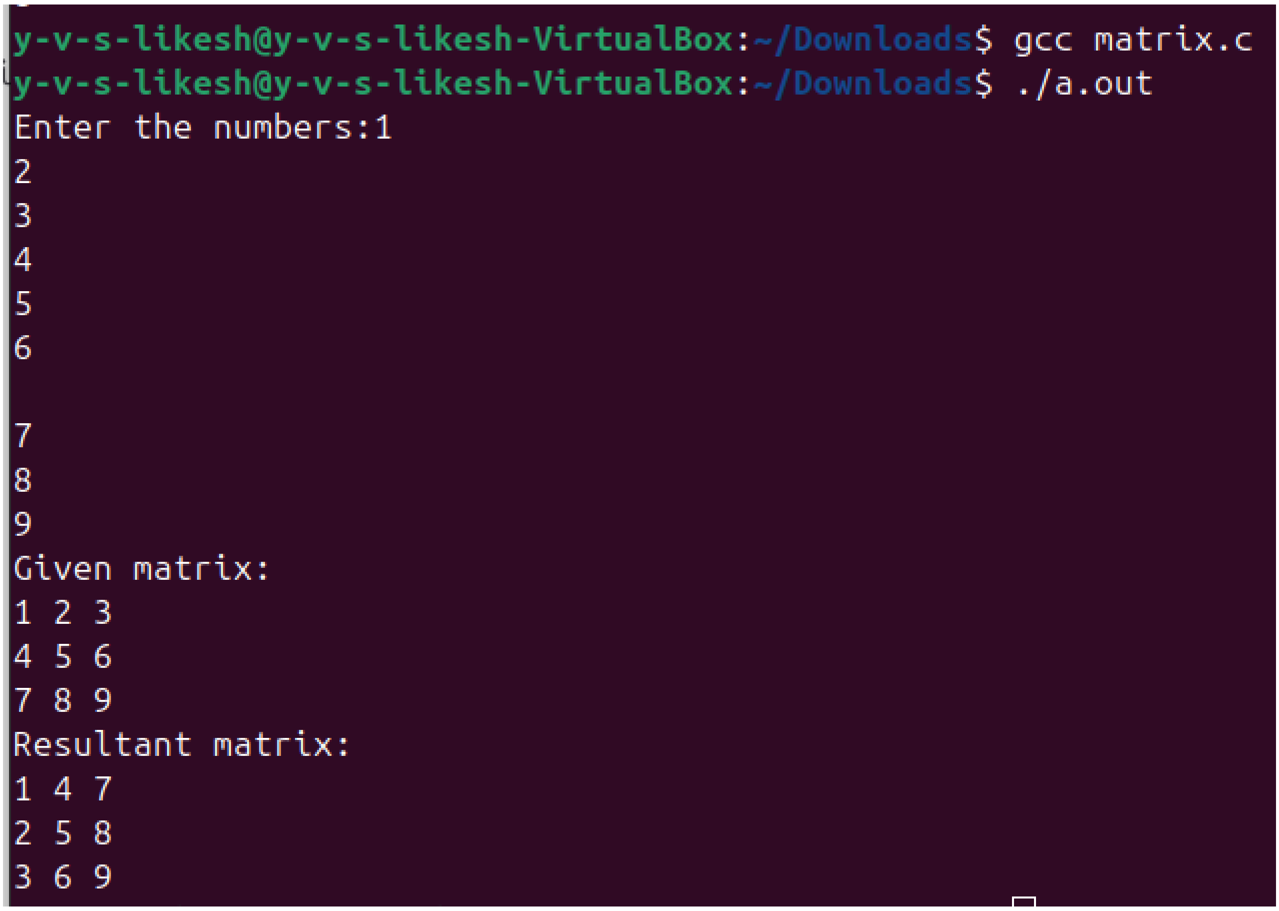
5) Write a program to find transpose of 3*3 matrix

Code:



```c
#include<stdio.h>
int main(){
int a[3][3],b[3][3];
printf("Enter the numbers:");
for(int i=0;i<3;i++){
 for(int j=0;j<3;j++){
  scanf("%d",&a[i][j]);
 }
}
for(int i=0;i<3;i++){
 for(int j=0;j<3;j++){
  b[i][j]=a[j][i];
 }
}
printf("Given matrix:\n");
for(int i=0;i<3;i++){
 for(int j=0;j<3;j++){
  printf("%d ",a[i][j]);
 }
 printf("\n");
}
printf("Resultant matrix:\n");
for(int i=0;i<3;i++){
 for(int j=0;j<3;j++){
  printf("%d ",b[i][j]);
 }
 printf("\n");
}
}
```

Output:



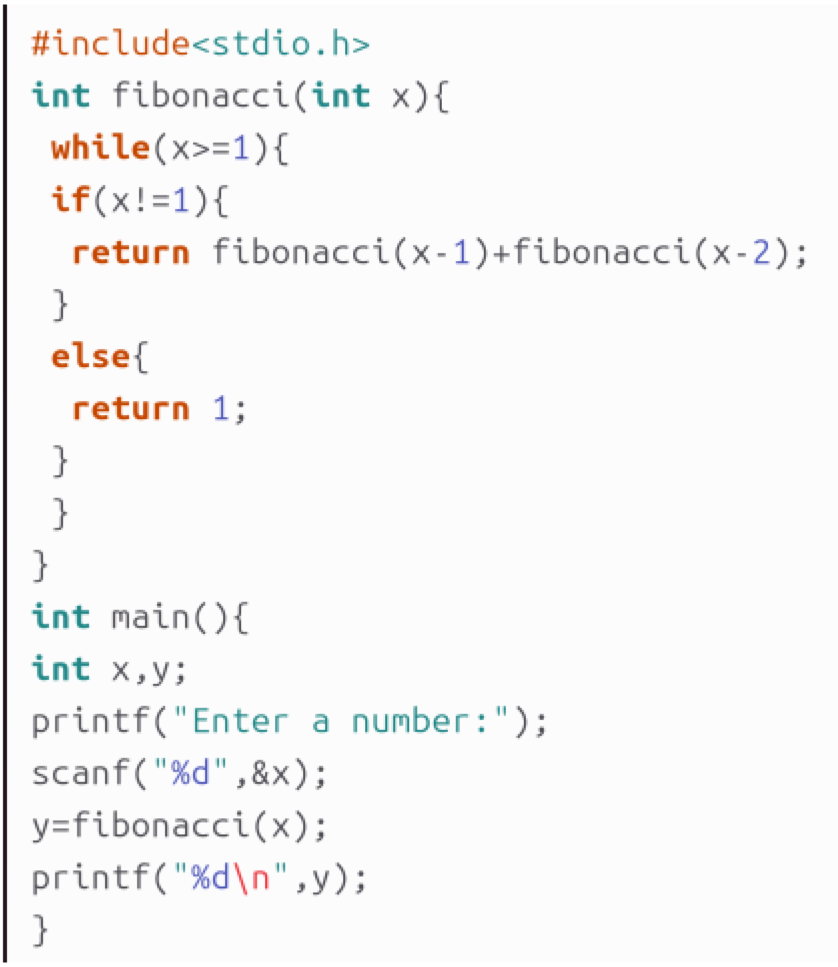
```
y-v-s-likesh@y-v-s-likesh-VirtualBox:~/Downloads$ gcc matrix.c
y-v-s-likesh@y-v-s-likesh-VirtualBox:~/Downloads$ ./a.out
Enter the numbers:1
2
3
4
5
6

7
8
9
Given matrix:
1 2 3
4 5 6
7 8 9
Resultant matrix:
1 4 7
2 5 8
3 6 9
```

Space Complexity:

The space complexity of this program is O(1) - constant space. Because the memory used does not grow with input size, the total space remains constant.

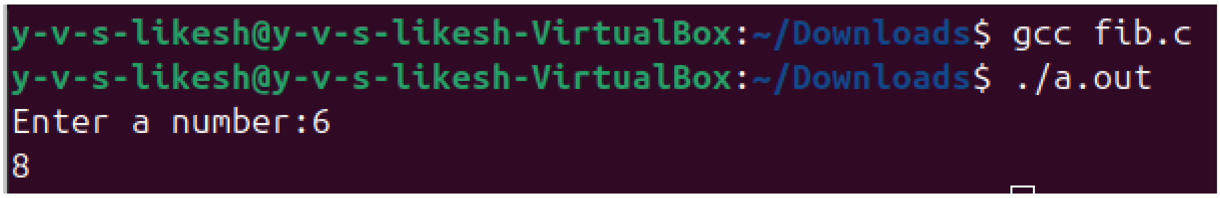
6) Write a program to find Fibonacci number at a given place

Code:



```
#include<stdio.h>
int fibonacci(int x){
 while(x>=1){
 if(x!=1){
  return fibonacci(x-1)+fibonacci(x-2);
 }
 else{
  return 1;
 }
 }
}
int main(){
int x,y;
printf("Enter a number:");
scanf("%d",&x);
y=fibonacci(x);
printf("%d\n",y);
}
```

Output:



```
y-v-s-likesh@y-v-s-likesh-VirtualBox:~/Downloads$ gcc fib.c
y-v-s-likesh@y-v-s-likesh-VirtualBox:~/Downloads$ ./a.out
Enter a number:6
8
```

Space Complexity:

The space complexity of this program is O(1) - constant space. There are no arrays, no recursion, and no dynamic memory allocation, so the memory usage does not depend on the number of Fibonacci terms printed.