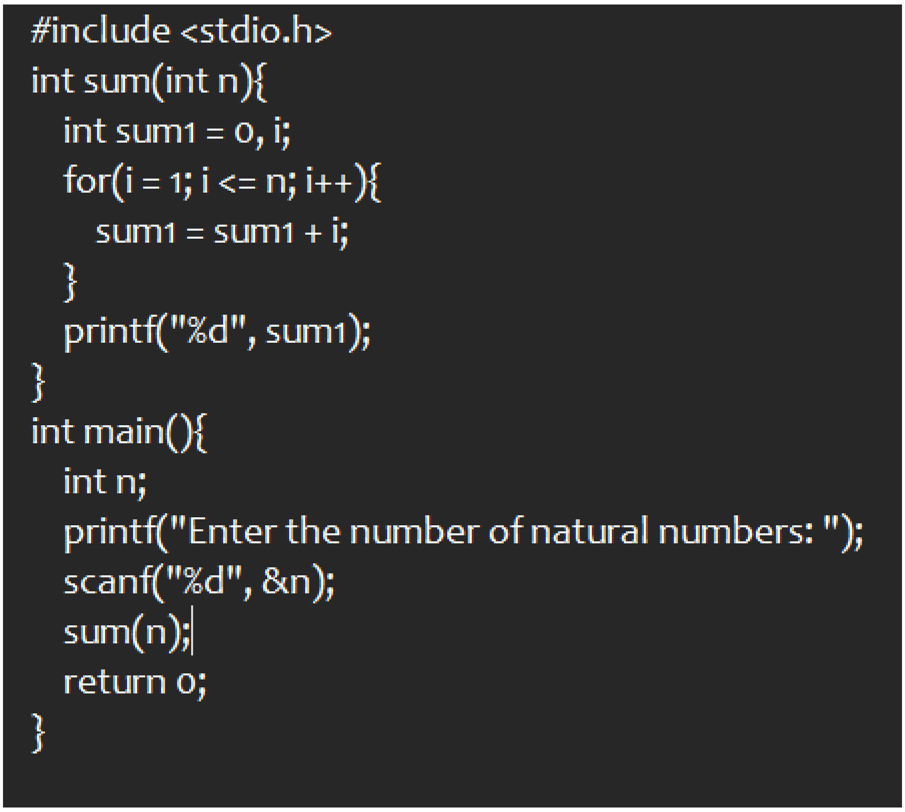
Design Analysis and Algorithm – Lab Work

Week 1

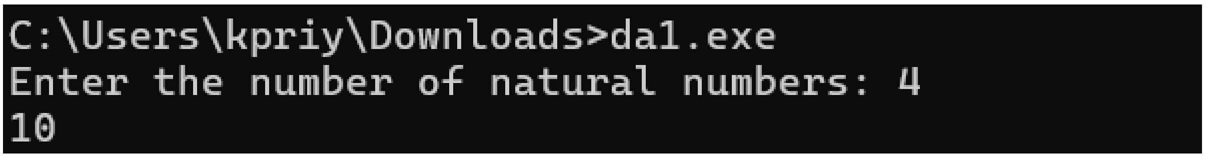
Question 1: Write a program to find sum of first n natural numbers using user defined function.

Code:



```
#include <stdio.h>
int sum(int n){
   int sum1 = 0, i;
   for(i = 1; i <= n; i++){
      sum1 = sum1 + i;
   }
   printf("%d", sum1);
}
int main(){
   int n;
   printf("Enter the number of natural numbers: ");
   scanf("%d", &n);
   sum(n);
   return 0;
}
```

Output:



```
C:\Users\kpriy\Downloads>da1.exe
Enter the number of natural numbers: 4
10
```

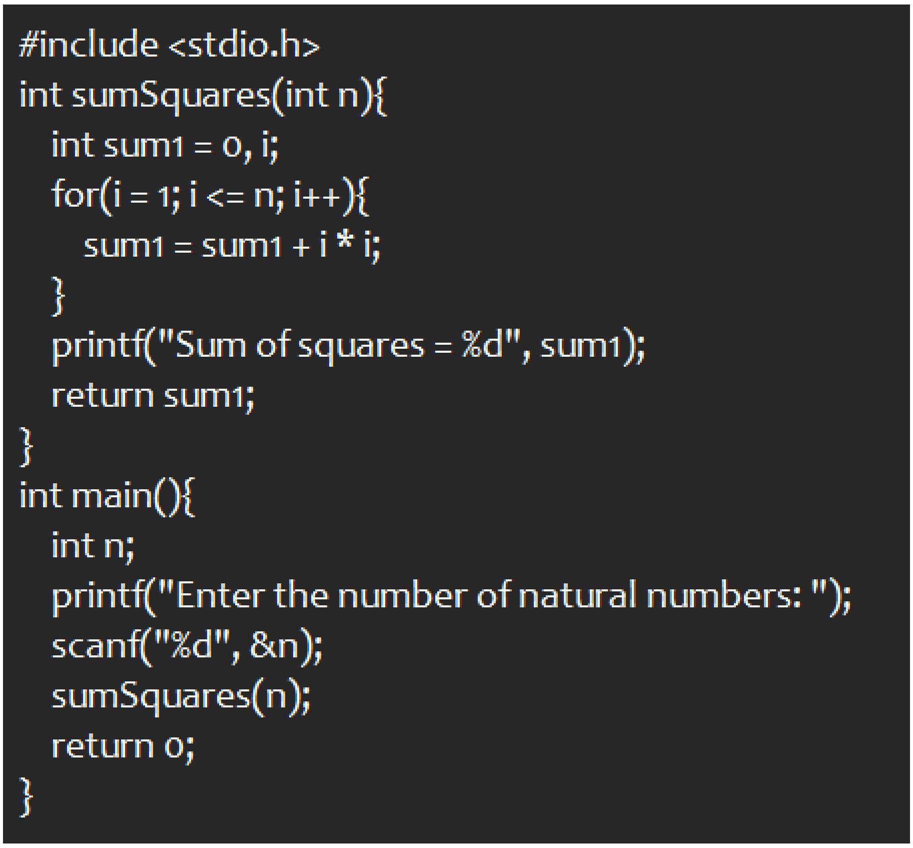
Space Complexity:

The space complexity of this code is O(1)[constant space].

The reason is that the memory does not increase even when n increases.

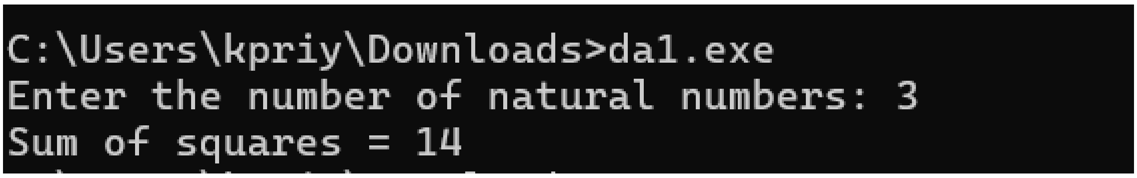
Question 2: Write a program to find sum of squares of the first n natural numbers.

Code:



```c
#include <stdio.h>
int sumSquares(int n){
   int sum1 = 0, i;
   for(i = 1; i <= n; i++){
      sum1 = sum1 + i * i;
   }
   printf("Sum of squares = %d", sum1);
   return sum1;
}
int main(){
   int n;
   printf("Enter the number of natural numbers: ");
   scanf("%d", &n);
   sumSquares(n);
   return 0;
}
```

Output:



```
C:\Users\kpriy\Downloads>da1.exe
Enter the number of natural numbers: 3
Sum of squares = 14
```

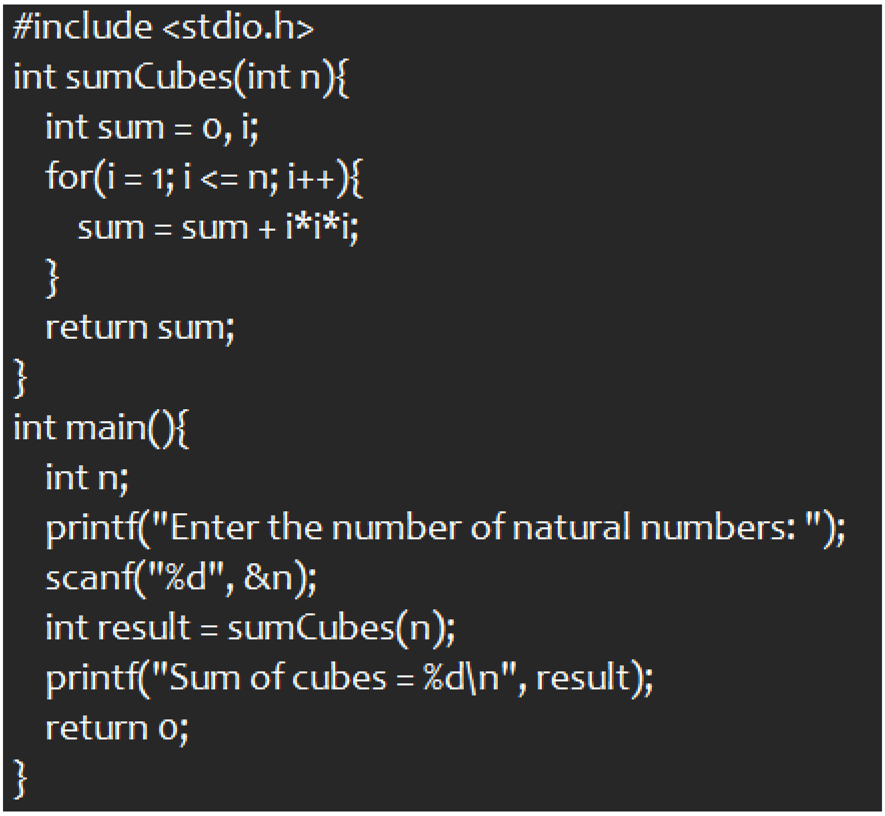
Space Complexity:

The space complexity of this code is O(1)[constant space].

The reason is that the memory does not increase even when n increases.

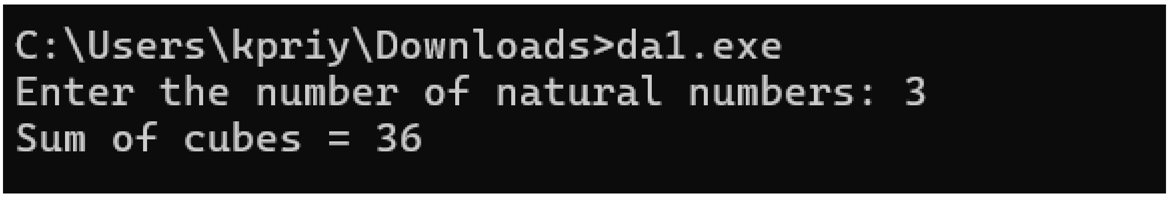
Question 3: Write a program to find sum of cubes of the first n natural numbers.

Code:



```c
#include <stdio.h>
int sumCubes(int n){
   int sum = 0, i;
   for(i = 1; i <= n; i++){
      sum = sum + i*i*i;
   }
   return sum;
}
int main(){
   int n;
   printf("Enter the number of natural numbers: ");
   scanf("%d", &n);
   int result = sumCubes(n);
   printf("Sum of cubes = %d\n", result);
   return 0;
}
```

Output:



```
C:\Users\kpriy\Downloads>da1.exe
Enter the number of natural numbers: 3
Sum of cubes = 36
```

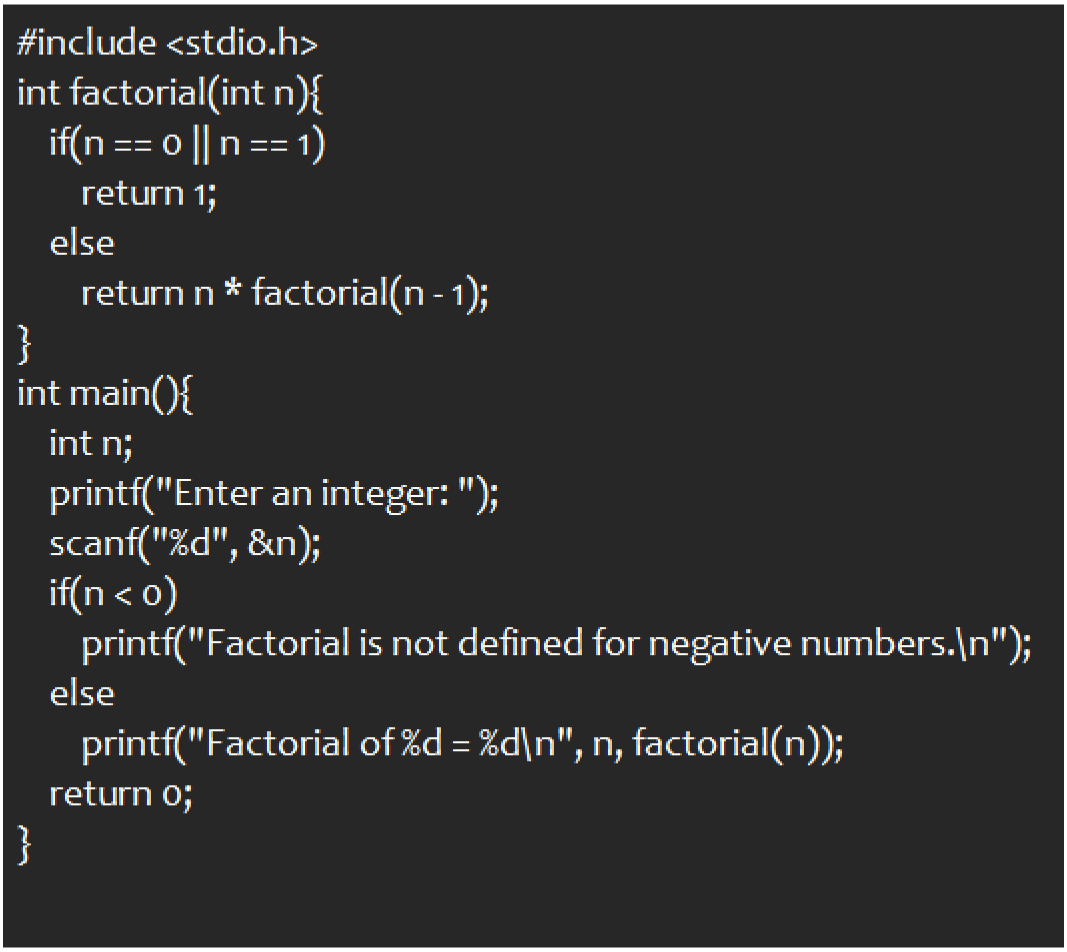
Space Complexity:

The space complexity of this code is O(1)[constant space].

The reason is that the memory does not increase even when n increases.

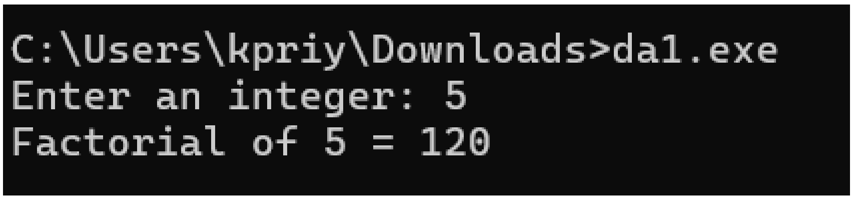
Question 4: Write a program to find the factorial of a given integer using recursion.

Code:



```c
#include <stdio.h>
int factorial(int n){
  if(n == 0 || n == 1)
    return 1;
  else
    return n * factorial(n - 1);
}
int main(){
  int n;
  printf("Enter an integer: ");
  scanf("%d", &n);
  if(n < 0)
    printf("Factorial is not defined for negative numbers.\n");
  else
    printf("Factorial of %d = %d\n", n, factorial(n));
  return 0;
}
```

Output:



```
C:\Users\kpriy\Downloads>da1.exe
Enter an integer: 5
Factorial of 5 = 120
```

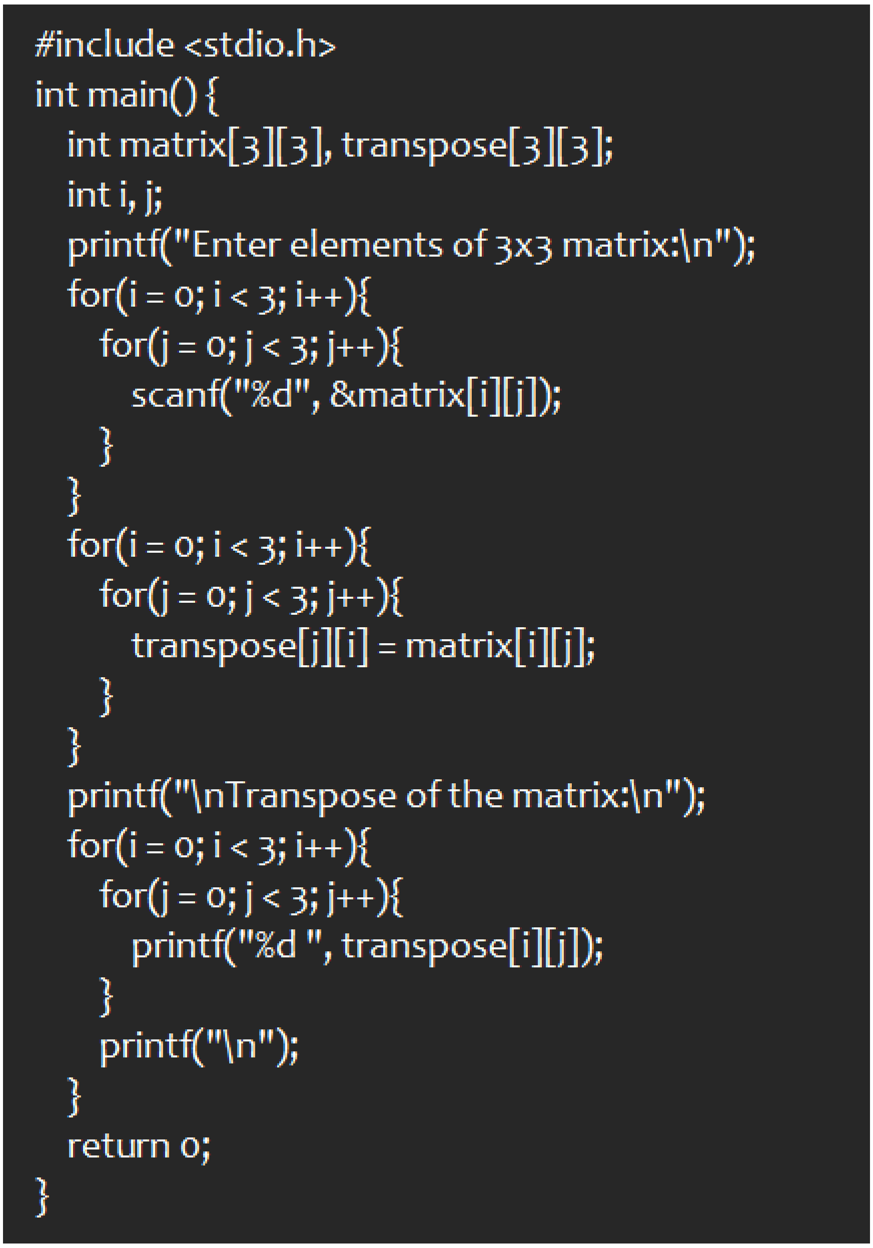
Space Complexity:

The space complexity of this code is O(n).

The reason is that the memory gets increased as we have used recursion in the function, so it depends upon n times.

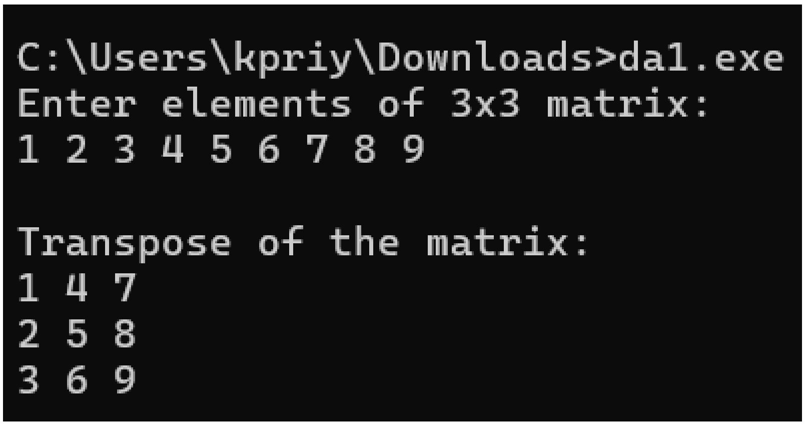
Question 5: Write a program for transposing a 3 x 3 matrix.

Code:



```c
#include <stdio.h>
int main() {
  int matrix[3][3], transpose[3][3];
  int i, j;
  printf("Enter elements of 3x3 matrix:\n");
  for(i = 0; i < 3; i++){
    for(j = 0; j < 3; j++){
      scanf("%d", &matrix[i][j]);
    }
  }
  for(i = 0; i < 3; i++){
    for(j = 0; j < 3; j++){
      transpose[j][i] = matrix[i][j];
    }
  }
  printf("\nTranspose of the matrix:\n");
  for(i = 0; i < 3; i++){
    for(j = 0; j < 3; j++){
      printf("%d ", transpose[i][j]);
    }
    printf("\n");
  }
  return 0;
}
```

Output:



```
C:\Users\kpriy\Downloads>da1.exe
Enter elements of 3x3 matrix:
1 2 3 4 5 6 7 8 9

Transpose of the matrix:
1 4 7
2 5 8
3 6 9
```

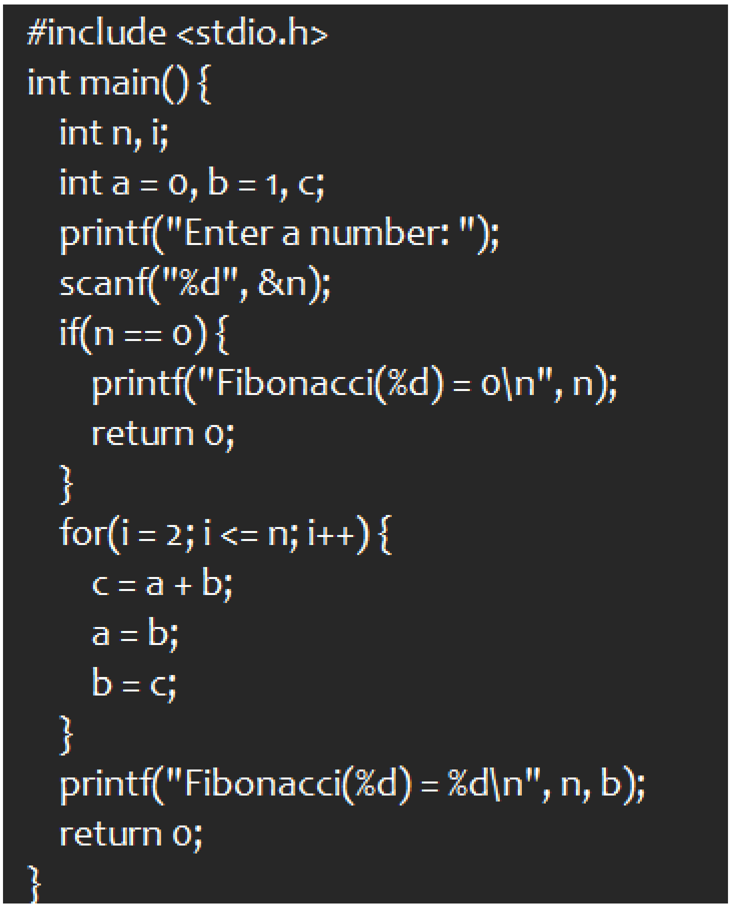
Space Complexity:

The space complexity of this code is O(1)[constant space].

The reason is that the memory does not increase even when n increases.

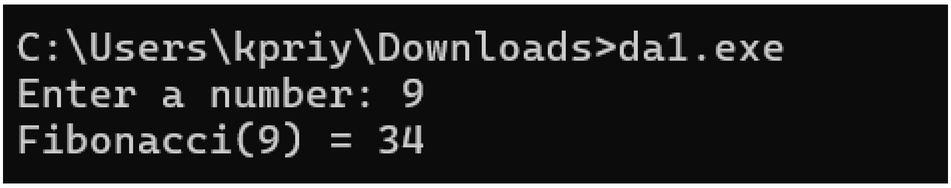
Question 6: Write a program to calculate Fibonacci of a number.

Code:



```
#include <stdio.h>
int main() {
   int n, i;
   int a = 0, b = 1, c;
   printf("Enter a number: ");
   scanf("%d", &n);
   if(n == 0) {
      printf("Fibonacci(%d) = 0\n", n);
      return 0;
   }
   for(i = 2; i <= n; i++) {
      c = a + b;
      a = b;
      b = c;
   }
   printf("Fibonacci(%d) = %d\n", n, b);
   return 0;
}
```

Output:



```
C:\Users\kpriy\Downloads>da1.exe
Enter a number: 9
Fibonacci(9) = 34
```

Space Complexity:

The space complexity of this code is O(1)[constant space].

The reason is that the memory does not increase even when n increases.