1. The Drumbeats of the Festival (Print 1 to n)

Story:

In a village festival, a drummer plays beats in increasing order.
He starts with beat 1 and goes up to beat n.

👉 Can you print the beats in order using recursion?

Input:

- Integer n (number of beats).

Output:

- Numbers from 1 to n separated by space.

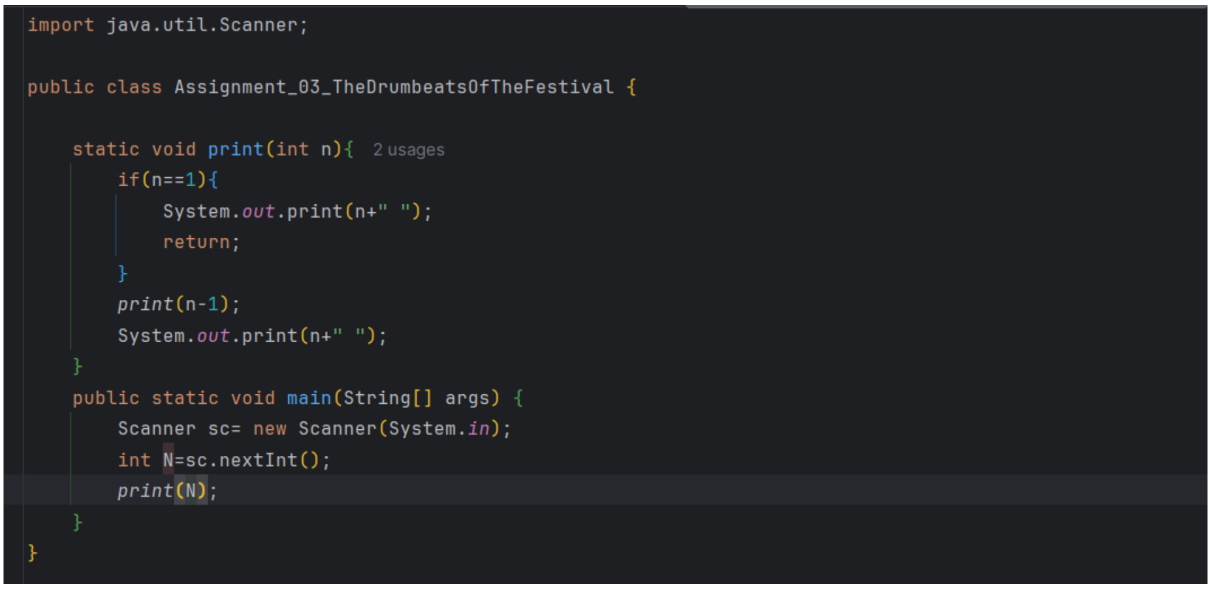
Constraints:

- 1 <= n <= 1000

Example:

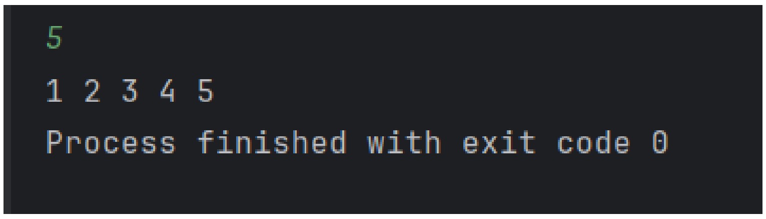
Input: 5
Output: 1 2 3 4 5

CODE :-



```java
import java.util.Scanner;

public class Assignment_03_TheDrumbeatsOfTheFestival {

    static void print(int n){
        if(n==1){
            System.out.print(n+" ");
            return;
        }
        print(n-1);
        System.out.print(n+" ");
    }
    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        int N=sc.nextInt();
        print(N);
    }
}
```

OUTPUT :-



```
5
1 2 3 4 5
Process finished with exit code 0
```

2. The Echo in the Cave (Print n to 1)

Story:
Inside a magical cave, a traveler shouts a number n.
The echo answers back in **descending order** down to 1.

👉 Print numbers from n to 1 using recursion.

Input:

- Integer n.

Output:

- Numbers from n to 1 separated by space.

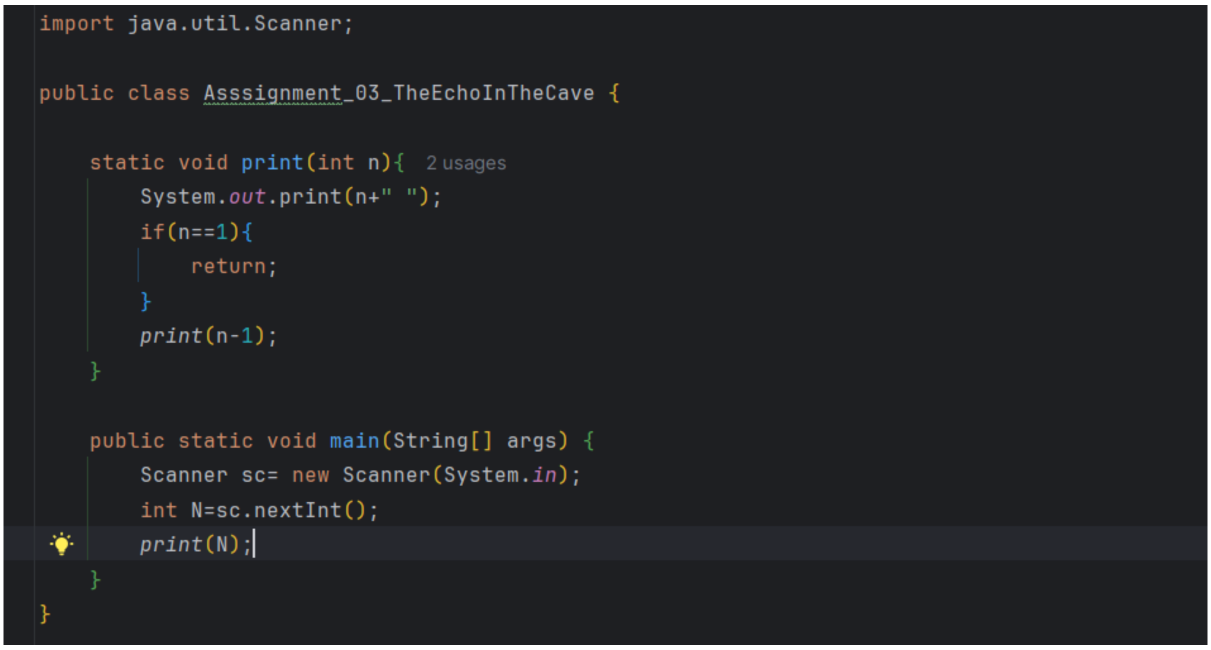
Constraints:

- 1 <= n <= 1000

Example:

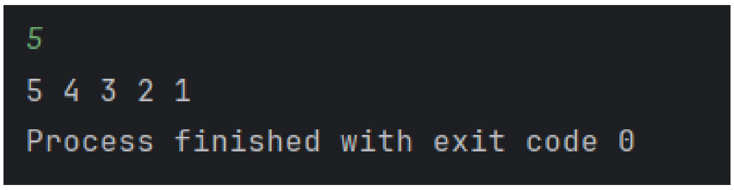
Input: 5
Output: 5 4 3 2 1

CODE :-



```
import java.util.Scanner;

public class Asssignment_03_TheEchoInTheCave {

    static void print(int n){  2 usages
        System.out.print(n+" ");
        if(n==1){
            return;
        }
        print(n-1);
    }

    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        int N=sc.nextInt();
        print(N);
    }
}
```

OUTPUT :-



```
5
5 4 3 2 1
Process finished with exit code 0
```

3. The King's Treasury (Sum of First n Numbers)

Story:
The King of Numberia has n treasure chests.
Each chest contains exactly the same number of coins as its position.
(Chest 1 has 1 coin, Chest 2 has 2 coins, … Chest n has n coins).

👉 Find the total coins using recursion.

Input:

- Integer n.

Output:

- The sum of numbers from 1 to n.

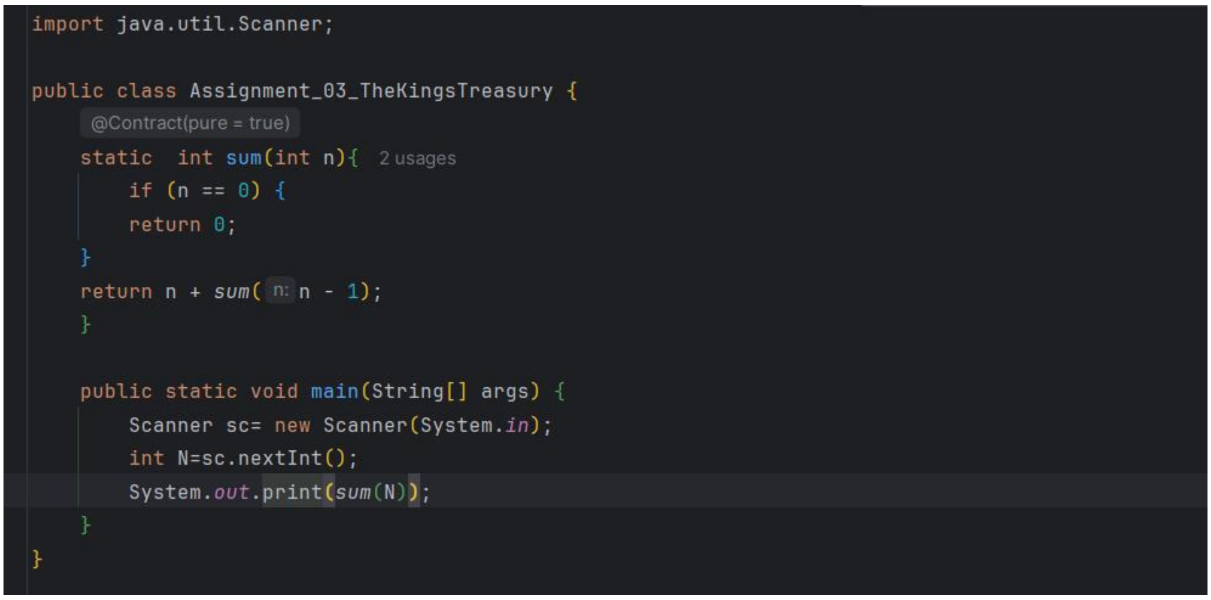
Constraints:

- 1 <= n <= 10^4

Example:

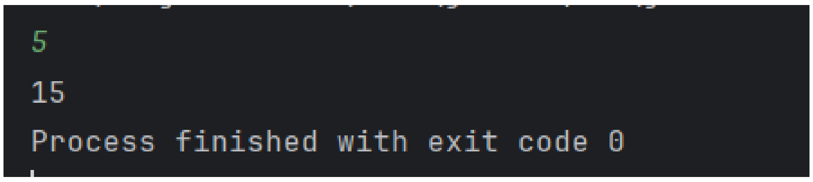
Input: 5
Output: 15
Explanation: 1+2+3+4+5 = 15

CODE :-



```
import java.util.Scanner;

public class Assignment_03_TheKingsTreasury {
    @Contract(pure = true)
    static  int sum(int n){  2 usages
        if (n == 0) {
        return 0;
    }
    return n + sum( n: n - 1);
    }

    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        int N=sc.nextInt();
        System.out.print(sum(N));
    }
}
```

OUTPUT :-



```
5
15
Process finished with exit code 0
```

4. The Wizard’s Mirror (Reverse String)

Story:
The wizard’s mirror reverses any word spoken into it.

👉 Reverse a string using recursion.

Input:

- String s.

Output:

- Reversed string.

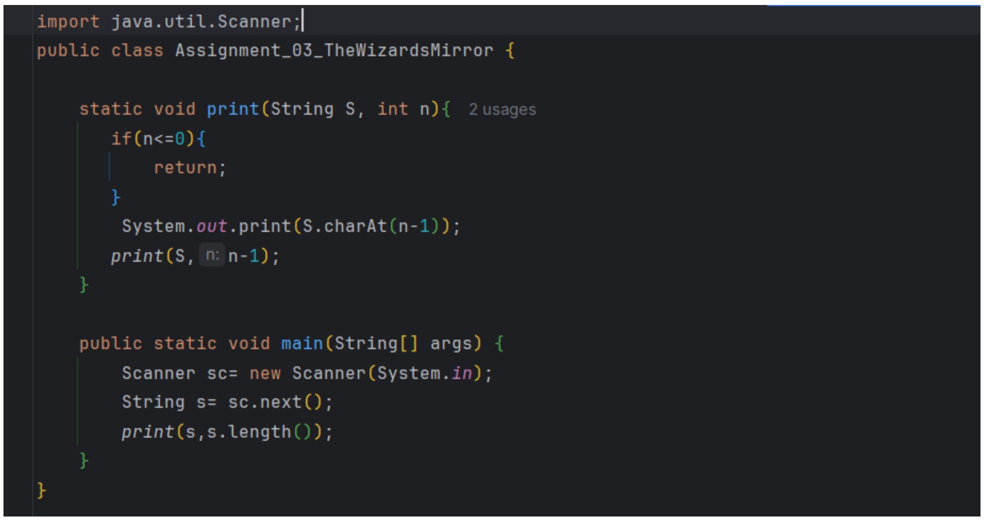
Constraints:

- 1 <= s.length <= 100

Example:

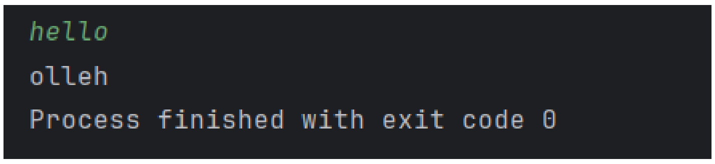
Input: hello
Output: olleh

CODE :-



```java
import java.util.Scanner;
public class Assignment_03_TheWizardsMirror {

    static void print(String S, int n){  2 usages
        if(n<=0){
            return;
        }
         System.out.print(S.charAt(n-1));
        print(S, n: n-1);
    }

    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        String s= sc.next();
        print(s,s.length());
    }
}
```

OUTPUT :-



```
hello
olleh
Process finished with exit code 0
```

5. The Treasure Boxes (Sum of Array)

Story:
A hero finds n treasure boxes, each with some coins.
He opens them one by one and counts the coins.

👉 Find the total coins using recursion.

Input:

- First line: integer n
- Second line: n integers (coins in each box).

Output:

- Sum of coins.

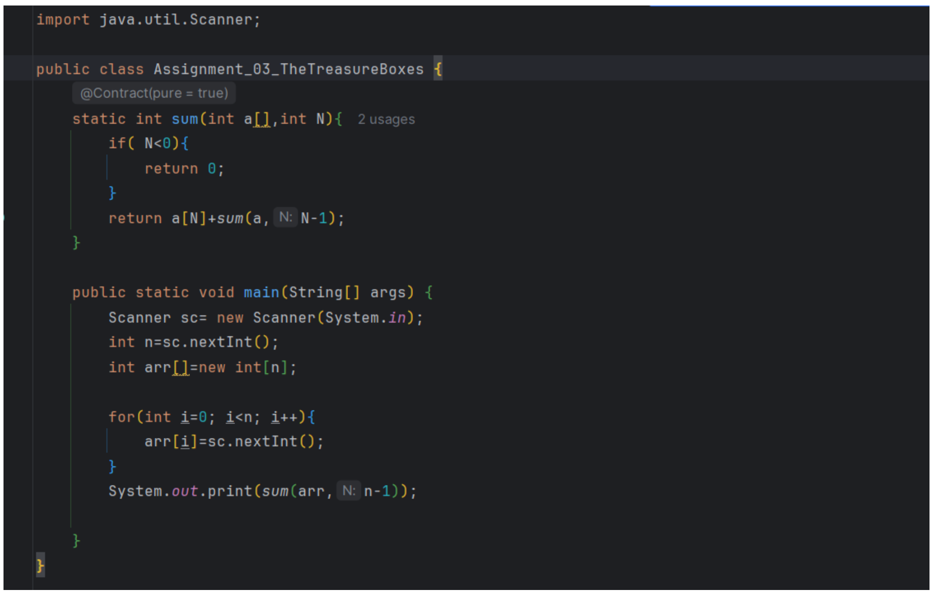
Constraints:

- 1 <= n <= 100
- 1 <= coins[i] <= 1000

Example:

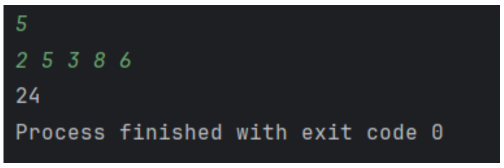
Input:
5
2 5 3 8 6
Output:
24

CODE :-



```java
import java.util.Scanner;

public class Assignment_03_TheTreasureBoxes {
    @Contract(pure = true)
    static int sum(int a[],int N){  2 usages
        if( N<0){
            return 0;
        }
        return a[N]+sum(a, N: N-1);
    }

    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        int n=sc.nextInt();
        int arr[]=new int[n];

        for(int i=0; i<n; i++){
            arr[i]=sc.nextInt();
        }
        System.out.print(sum(arr, N: n-1));

    }
}
```

OUTPUT :-



```
5
2 5 3 8 6
24
Process finished with exit code 0
```

6. The Traveler's Steps (Climbing Stairs)

Story:
A traveler must climb a staircase with n magical steps.
He can climb **1 step or 2 steps at a time**.

👉 Find the number of distinct ways to reach the top using recursion.

Input:

- Integer n.

Output:

- Number of ways to climb.

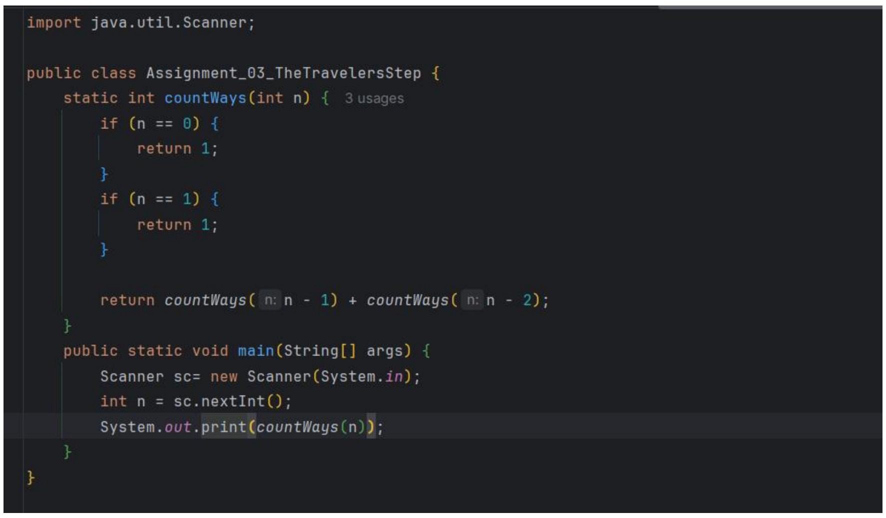
Constraints:

- 1 <= n <= 30

Example:

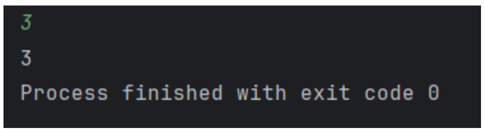
Input: 3
Output: 3
Explanation: {1+1+1, 1+2, 2+1}

CODE :-



```
import java.util.Scanner;

public class Assignment_03_TheTravelersStep {
    static int countWays(int n) {  3 usages
        if (n == 0) {
            return 1;
        }
        if (n == 1) {
            return 1;
        }

        return countWays( n: n - 1) + countWays( n: n - 2);
    }
    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        int n = sc.nextInt();
        System.out.print(countWays(n));
    }
}
```

OUTPUT :-



```
3
3
Process finished with exit code 0
```

7. The Princess's Lock (Factorial)

Story:
The princess is locked behind n magical locks.
She can only unlock them in **every possible order**.

👉 How many ways can she open them? (factorial)

Input:

- Integer n.

Output:

- Factorial of n.

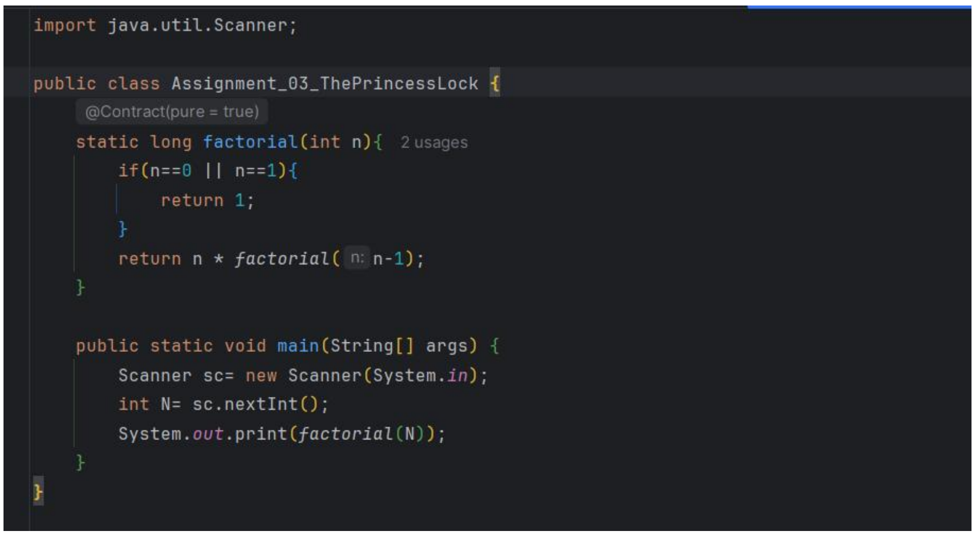
Constraints:

- $1 <= n <= 12$

Example:

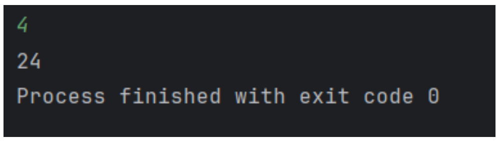
Input: 4
Output: 24
Explanation: $4! = 4×3×2×1$

CODE :-



```java
import java.util.Scanner;

public class Assignment_03_ThePrincessLock {
    @Contract(pure = true)
    static long factorial(int n){  2 usages
        if(n==0 || n==1){
            return 1;
        }
        return n * factorial( n: n-1);
    }

    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        int N= sc.nextInt();
        System.out.print(factorial(N));
    }
}
```

OUTPUT :-



```
4
24
Process finished with exit code 0
```

8 . The Rabbit's Family (Fibonacci)

Story:
In a magical forest, rabbits grow as:

- Month 1 → 1 rabbit
- Month 2 → 1 rabbit
- From Month 3 → rabbits = sum of previous two months.

👉 Find number of rabbits after n months.

Input:

- Integer n.

Output:

- Fibonacci number at month n.

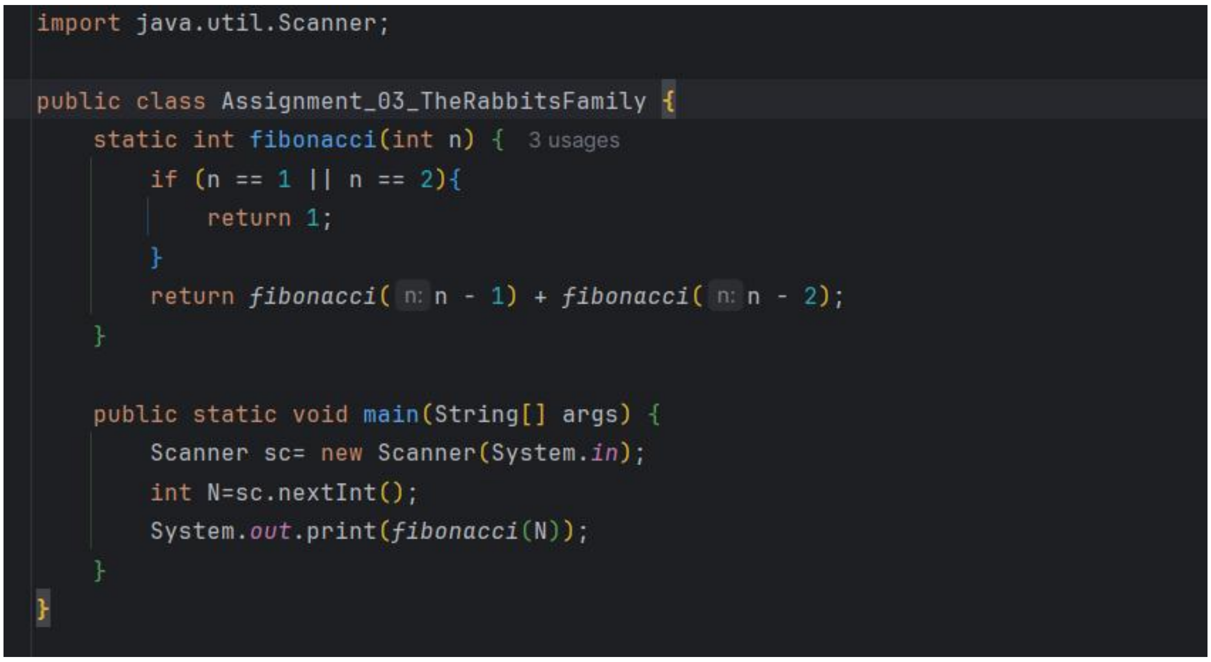
Constraints:

- 1 <= n <= 40

Example:

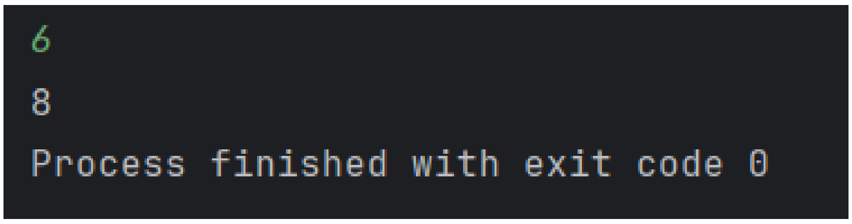
Input: 6
Output: 8
Explanation: 1,1,2,3,5,8

CODE :-



```
import java.util.Scanner;

public class Assignment_03_TheRabbitsFamily {
    static int fibonacci(int n) {  3 usages
        if (n == 1 || n == 2){
            return 1;
        }
        return fibonacci( n: n - 1) + fibonacci( n: n - 2);
    }

    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        int N=sc.nextInt();
        System.out.print(fibonacci(N));
    }
}
```

OUTPUT :-



```
6
8
Process finished with exit code 0
```