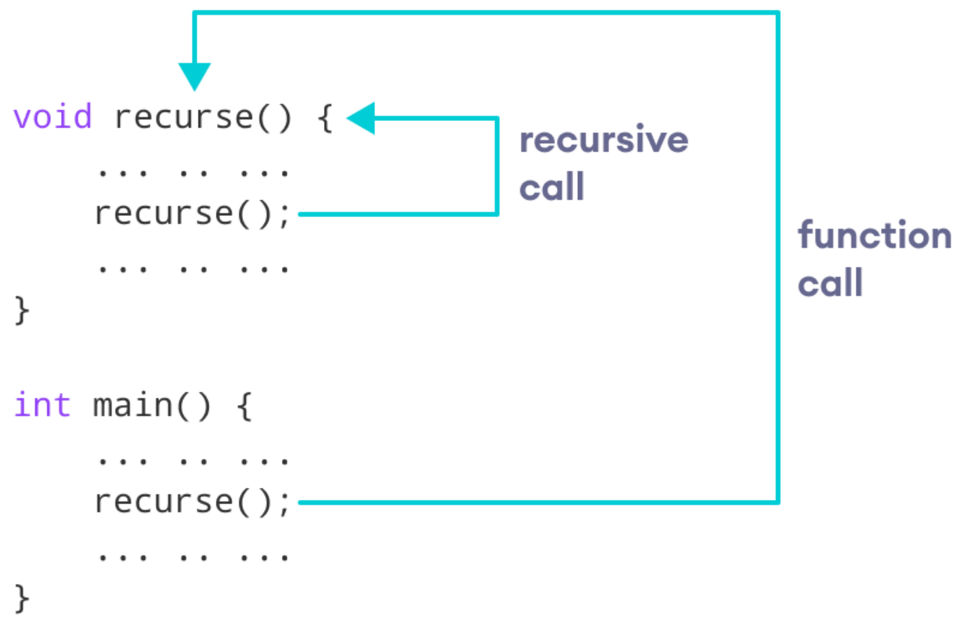
Data Structures and Algorithms

Lecture 22: Recursion

Recursion

A function that calls itself is known as a recursive function. And, this technique is known as recursion.



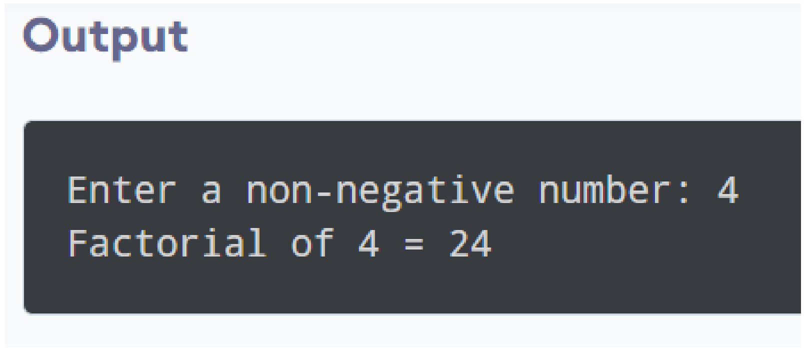
```
void recurse() {
    ... .. ...
    recurse();
    ... .. ...
}

int main() {
    ... .. ...
    recurse();
    ... .. ...
}
```

The recursion continues until some condition is met.

Note: To prevent infinite recursion, if...else statement (or similar approach) can be used where one branch makes the recursive call and the other doesn't.

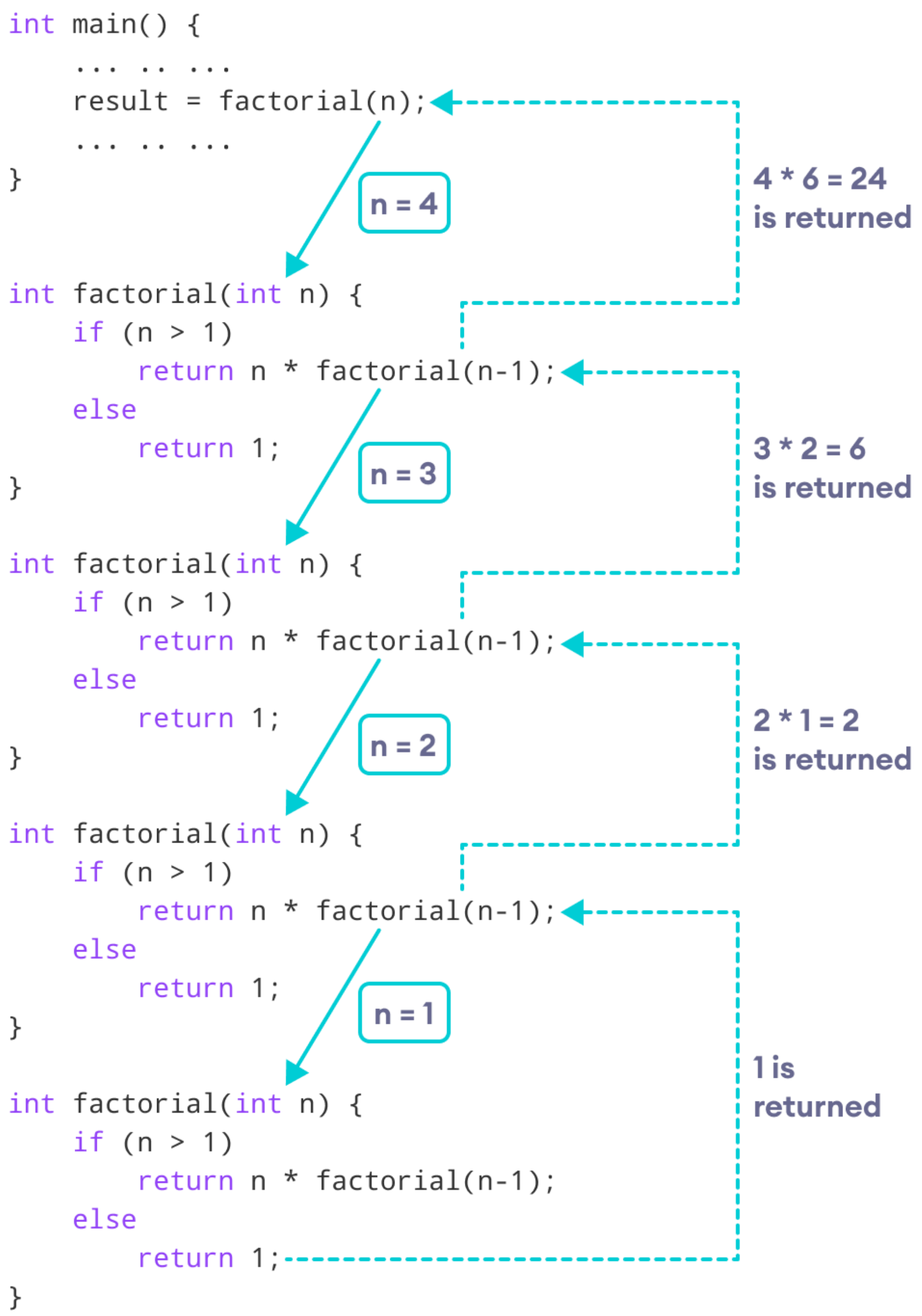
Factorial of a number using recursion

```
#include <iostream>
using namespace std;
int factorial(int);
int main() {
   int n, result;
   cout << "Enter a non-negative number: ";
   cin >> n;
   result = factorial(n);
   cout << "Factorial of " << n << " = " << result;
   return 0;
}
    int factorial (int n)
    {
       if (n > 1)
       {
          return n * factorial (n - 1);
       } else {
          return 1;
       }
    }
```



Output

```
Enter a non-negative number: 4
Factorial of 4 = 24
```



```
int main() {
    ... .. ...
    result = factorial(n);
    ... .. ...
}

int factorial(int n) {
    if (n > 1)
        return n * factorial(n-1);
    else
        return 1;
}

int factorial(int n) {
    if (n > 1)
        return n * factorial(n-1);
    else
        return 1;
}

int factorial(int n) {
    if (n > 1)
        return n * factorial(n-1);
    else
        return 1;
}

int factorial(int n) {
    if (n > 1)
        return n * factorial(n-1);
    else
        return 1;
}
```

```
return 5 * factorial(4) = 120
  └── return 4 * factorial(3) = 24
        └── return 3 * factorial(2) = 6
              └── return 2 * factorial(1) = 2
                    └── return 1 * factorial(0) = 1
```

Recursion

Advantages of Recursion

- It makes our code shorter and cleaner.
- Recursion is required in problems concerning data structures and advanced algorithms, such as Graph and Tree Traversal.

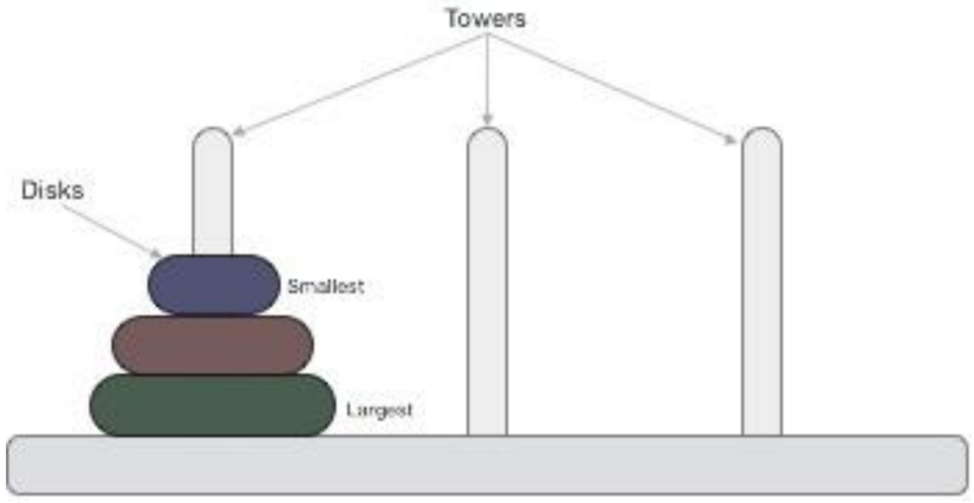
Disadvantages of Recursion

- It takes a lot of stack space compared to an iterative program.
- It uses more processor time.
- It can be more difficult to debug compared to an equivalent iterative program.

Tower of Hanoi Problem

Tower of Hanoi, is a mathematical puzzle which consists of three towers (pegs) and more than one rings is as depicted −

These rings are of different sizes and stacked upon in an ascending order, i.e. the smaller one sits over the larger one. There are other variations of the puzzle where the number of disks increase, but the tower count remains the same.



Tower of Hanoi Problem

- Rules

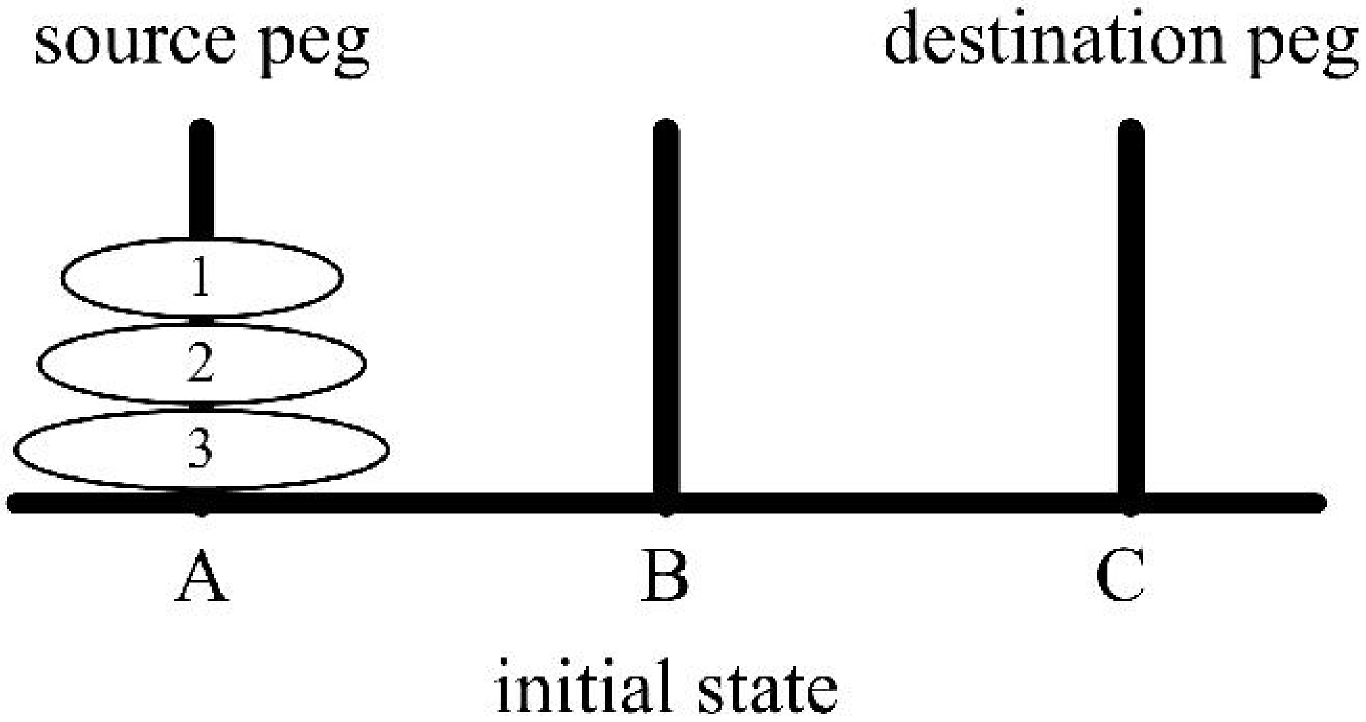
The mission is to move all the disks to some another tower without violating the sequence of arrangement.

A few rules to be followed for Tower of Hanoi are −

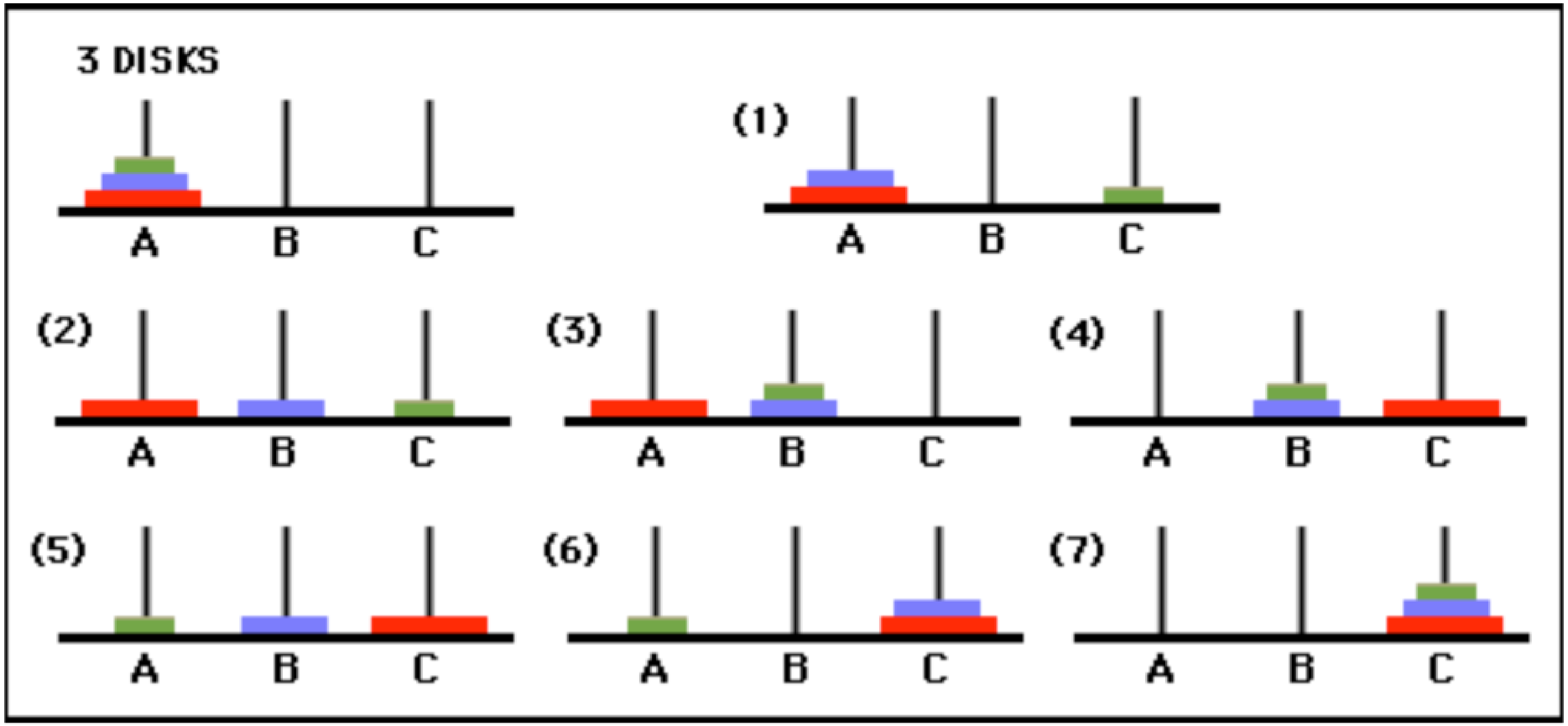
- Only one disk can be moved among the towers at any given time.
- Only the "top" disk can be removed.
- No large disk can sit over a small disk.

The minimal number of moves required to solve the Tower of Hanoi puzzle of n disks would be (2^n) − 1.

Tower of Hanoi Problem



Tower of Hanoi Problem

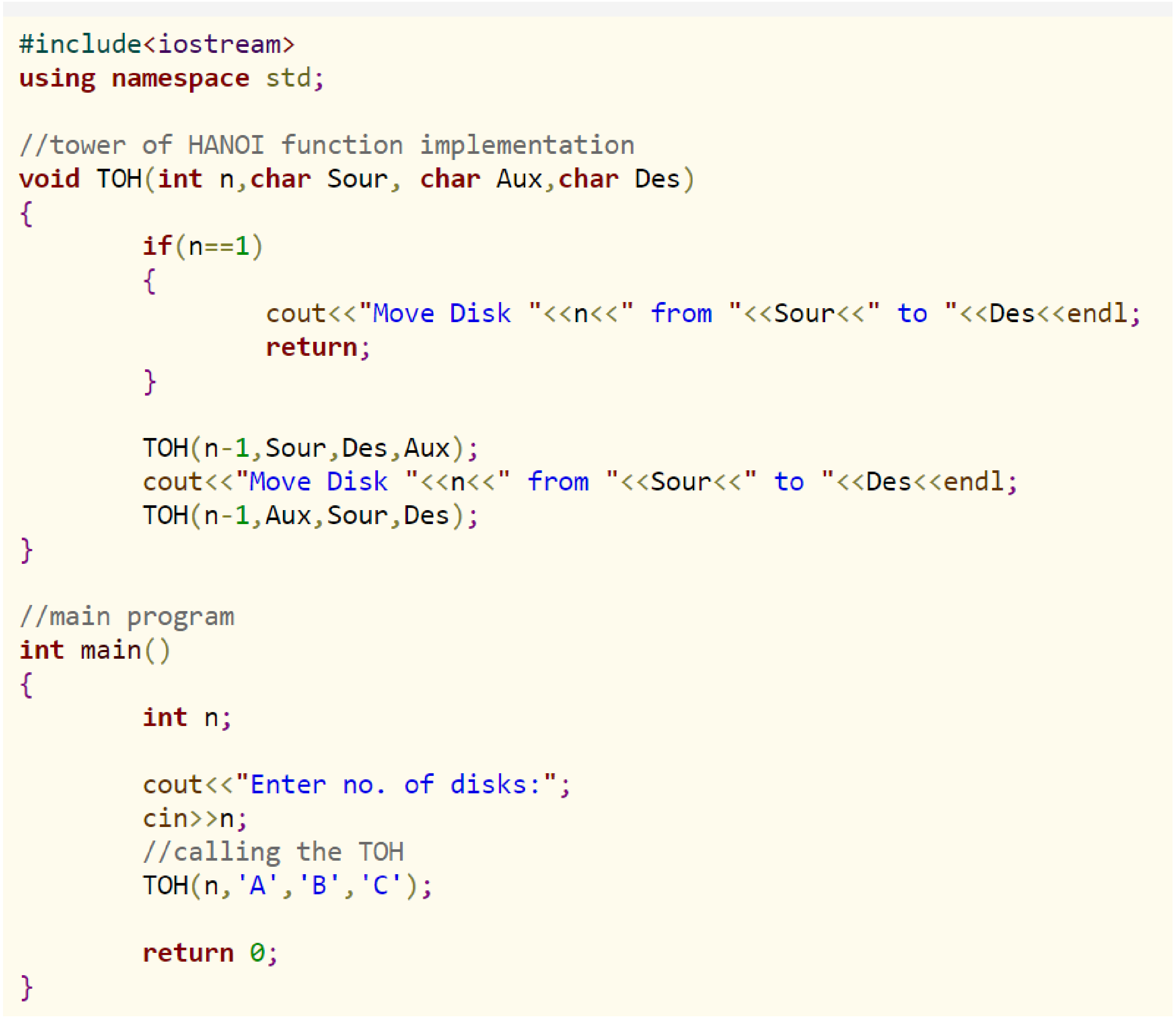


Tower of Hanoi Problem

Algorithm

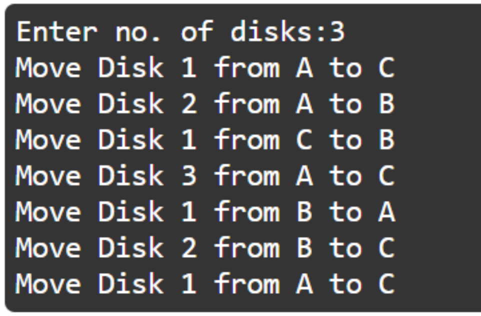
```
TOH( n,  Sour, Aux , Des)
If(n=1)
    Write ("Move Disk “, n ," from ", Sour ," to ",Des)
Else
    TOH(n-1,Sour,Des,Aux);
    Write ("Move Disk “, n ," from ", Sour ," to ",Des)
    TOH(n-1,Aux,Sour,Des);
END
```

Tower of Hanoi Problem



```cpp
#include<iostream>
using namespace std;

//tower of HANOI function implementation
void TOH(int n,char Sour, char Aux,char Des)
{
        if(n==1)
        {
                cout<<"Move Disk "<<n<<" from "<<Sour<<" to "<<Des<<endl;
                return;
        }

        TOH(n-1,Sour,Des,Aux);
        cout<<"Move Disk "<<n<<" from "<<Sour<<" to "<<Des<<endl;
        TOH(n-1,Aux,Sour,Des);
}

//main program
int main()
{
        int n;

        cout<<"Enter no. of disks:";
        cin>>n;
        //calling the TOH
        TOH(n,'A','B','C');

        return 0;
}
```

Output



```
Enter no. of disks:3
Move Disk 1 from A to C
Move Disk 2 from A to B
Move Disk 1 from C to B
Move Disk 3 from A to C
Move Disk 1 from B to A
Move Disk 2 from B to C
Move Disk 1 from A to C
```