

Non recursive TOH

```
#include <iostream>
```

```
#include<stack>
```

```
#include <cmath>
```

Using namespace std;

```
Void movedisk(char from,char to,stack<int>&src,stack<int>&des){
```

```
    If(src.empty() && des.empty())
```

```
    Return;
```

```
    If(src.empty()){
```

```
        Int disk=des.top();
```

```
        Des.pop();
```

```
        Src.push(disk);
```

```
        Cout<<"move disk "<<disk<<"from"<<to<<"to"<<from<<endl;
```

```
    }else if(des.empty()){
```

```
        Int disk=src.top();
```

```
        Src.pop();
```

```
        Des.push(disk);
```

```
        Cout<<"move disk "<<disk<<"from"<<from<<"to"<<to<<endl;
```

```
    }
```

```
    Else if(src.top()>des.top()){
```

```
        Int disk=des.top();
```

```
        Des.pop();
```

```
        Src.push(disk);
```

```
        Cout<<"move disk "<<disk<<"from"<<from<<"to"<<to<<endl;
```

```

    }else{
        Int disk=src.top();
        Src.pop();
        Des.push(disk);
        Cout<<"move disk "<<disk<<"from"<<from<<"to"<<to<<endl;
    }

```

```

}

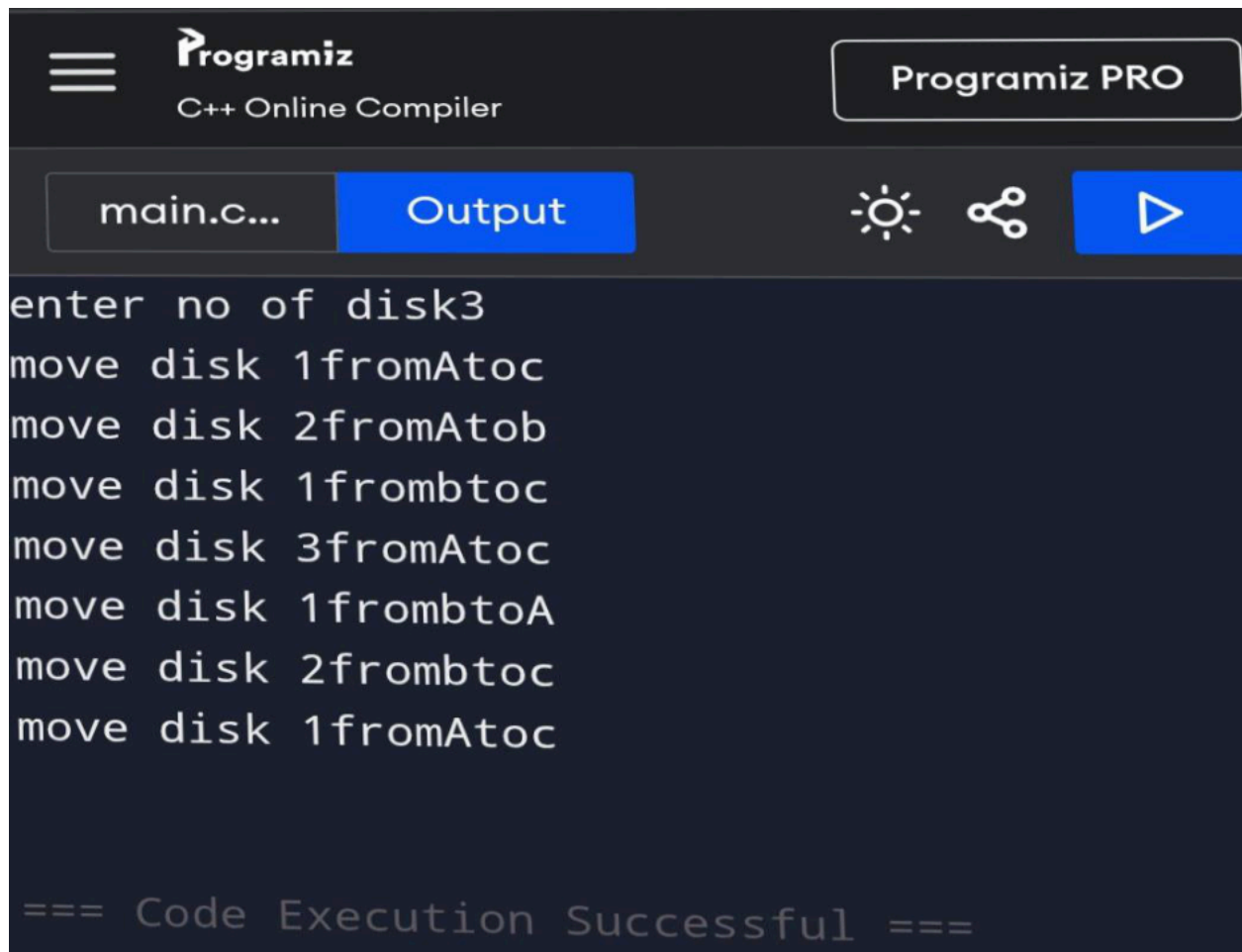
```

```

Int main() {
    Int n;
    Cout<<"enter no of disk";
    Cin>> n;
    Stack<int>src,aux,des;
    Int totalmoves=pow(2,n)-1;
    For(int i=n;i>=1;i--){
        Src.push(i);
    }
    Char s='A',a='b',d='c';
    If(n%2==0){
        Swap(a,d);
    }
}

```

```
For(int i=1;i<=totalmoves;i++){  
    If(i%3==1){  
        Movedisk(s,d,src,des);  
    }else if(i%3==2){  
        Movedisk(s,a,src,aux);  
    }else {  
        Movedisk(a,d,aux,des);  
    }  
}  
}  
Return 0;  
}
```



The screenshot shows the Programiz C++ Online Compiler interface. At the top, there is a logo for Programiz and the text "C++ Online Compiler". On the right, there is a button labeled "Programiz PRO". Below the header, there is a tab labeled "main.c..." and a blue button labeled "Output". To the right of the "Output" button are three icons: a sun, a share icon, and a play button. The main area displays the output of the program, which is a sequence of moves for the Tower of Hanoi puzzle. The output is as follows:

```
enter no of disk3
move disk 1fromAtoc
move disk 2fromAtob
move disk 1frombtoc
move disk 3fromAtoc
move disk 1frombtoA
move disk 2frombtoc
move disk 1fromAtoc

=== Code Execution Successful ===
```

Recursive TOH

```
#include <iostream>
```

```
Void towerOfHanoi(int n, char source, char destination, char auxiliary) {
```

```
    If (n == 1) {
```

```

        Std::cout << "Move disk 1 from " << source << " to " << destination << std::endl;

        Return;

    }

    towerOfHanoi(n - 1, source, auxiliary, destination);

    std::cout << "Move disk " << n << " from " << source << " to " << destination << std::endl;

    towerOfHanoi(n - 1, auxiliary, destination, source);

}

Int main() {

    Int num_disks;

    Std::cout << "Enter the number of disks: ";

    Std::cin >> num_disks;

    towerOfHanoi(num_disks, 'A', 'C', 'B');

    return 0;

}

```

main.c...

Output

Enter the number of disks: 4

Move disk 1 from A to B

Move disk 2 from A to C

Move disk 1 from B to C

Move disk 3 from A to B

Move disk 1 from C to A

Move disk 2 from C to B

Move disk 1 from A to B

Move disk 4 from A to C

Move disk 1 from B to C

Move disk 2 from B to A

Move disk 1 from C to A

Move disk 3 from B to C

Move disk 1 from A to B

Move disk 2 from A to C

Move disk 1 from B to C

Stack implementation

```
#include <iostream>

using namespace std;

#define MAX 5 // Maximum size of stack

class Stack {

    int arr[MAX];

    int top;

public:

    Stack() { top = -1; }

    // Push operation

    void push(int x) {

        if (top == MAX - 1) {

            cout << "Stack Overflow\n";

            return;

        }

        arr[++top] = x;

        cout << x << " pushed\n";

    }
```

// Pop operation

```
void pop() {  
    if (top == -1) {  
        cout << "Stack Underflow\n";  
        return;  
    }  
    cout << arr[top--] << " popped\n";  
}
```

// Peek operation

```
int peek() {  
    if (top == -1) {  
        cout << "Stack is Empty\n";  
        return -1;  
    }  
    return arr[top];  
}
```

// Check empty

```
bool isEmpty() {  
    return (top == -1);  
}  
};
```



```
int main() {  
    Stack s;  
  
    s.push(10);  
    s.push(20);  
    s.push(30);  
  
    cout << "Top element: " << s.peek() << endl;  
  
    s.pop();  
    s.pop();  
  
    cout << "Is stack empty? " << (s.isEmpty() ? "Yes" : "No") << endl;  
  
    return 0;  
}
```

main.c...

Output

10 pushed

20 pushed

30 pushed

Top element: 30

30 popped

20 popped

Is stack empty? No