MA144: Problem Solving and Computer Programming

Lecture-18

Functions-2 (Recursion)

Recursion

Recursion is a situation where a function calls itself.

Finding factorial of a positive integer.

```cpp
#include<iostream>
using namespace std;
long factorial(int);
int main()
{   int n;
    long fact;
    cout<<" enter a positive integer: ";
    cin>>n;
    fact=factorial(n);
    cout<<"Factorial of "<<n<<" is "<<fact;
    return 0;
}
long factorial(int n)
{
    if(n==0)
     return 1;
    return (n*factorial(n-1));
 }
```

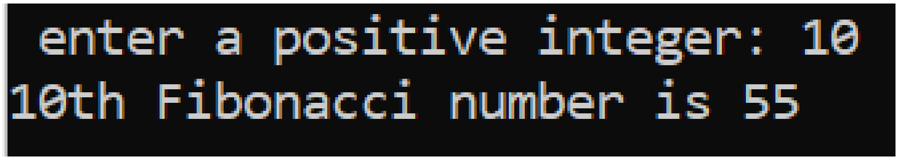
Terminating condition

Recursive function call

```
 enter a positive integer: 10
Factorial of 10 is 3628800
```

Finding n^{th} Fibonacci Number

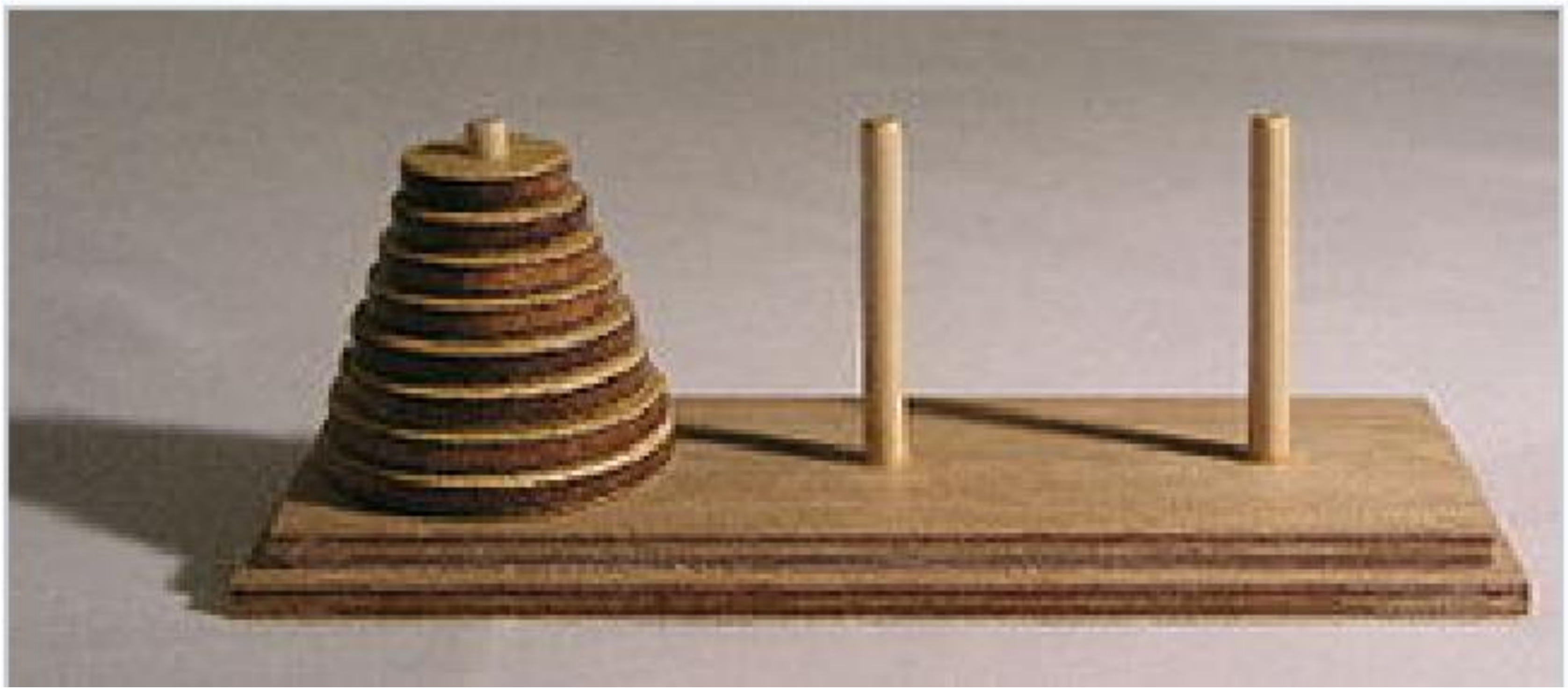
```cpp
#include<iostream>
using namespace std;
long nth_fibonacci(int);
int main()
{   int n;
    long fibo;
    cout<<" enter a positive integer: ";
    cin>>n;
    fibo=nth_fibonacci(n);
    cout<<n<<"th Fibonacci number is "<<fibo;
    return 0;
}
long nth_fibonacci(int n)
{
    if(n==1|| n==2)
     return 1;
    return (nth_fibonacci(n-1)+nth_fibonacci(n-2));
 }
```



```
 enter a positive integer: 10
10th Fibonacci number is 55
```

Tower of Hanoi Puzzle

The Tower of Hanoi, consists of three pegs mounted on a board together with disks of different sizes. Initially these disks are placed on the first peg in order of size, with the largest on the bottom (as shown in the following Figure).



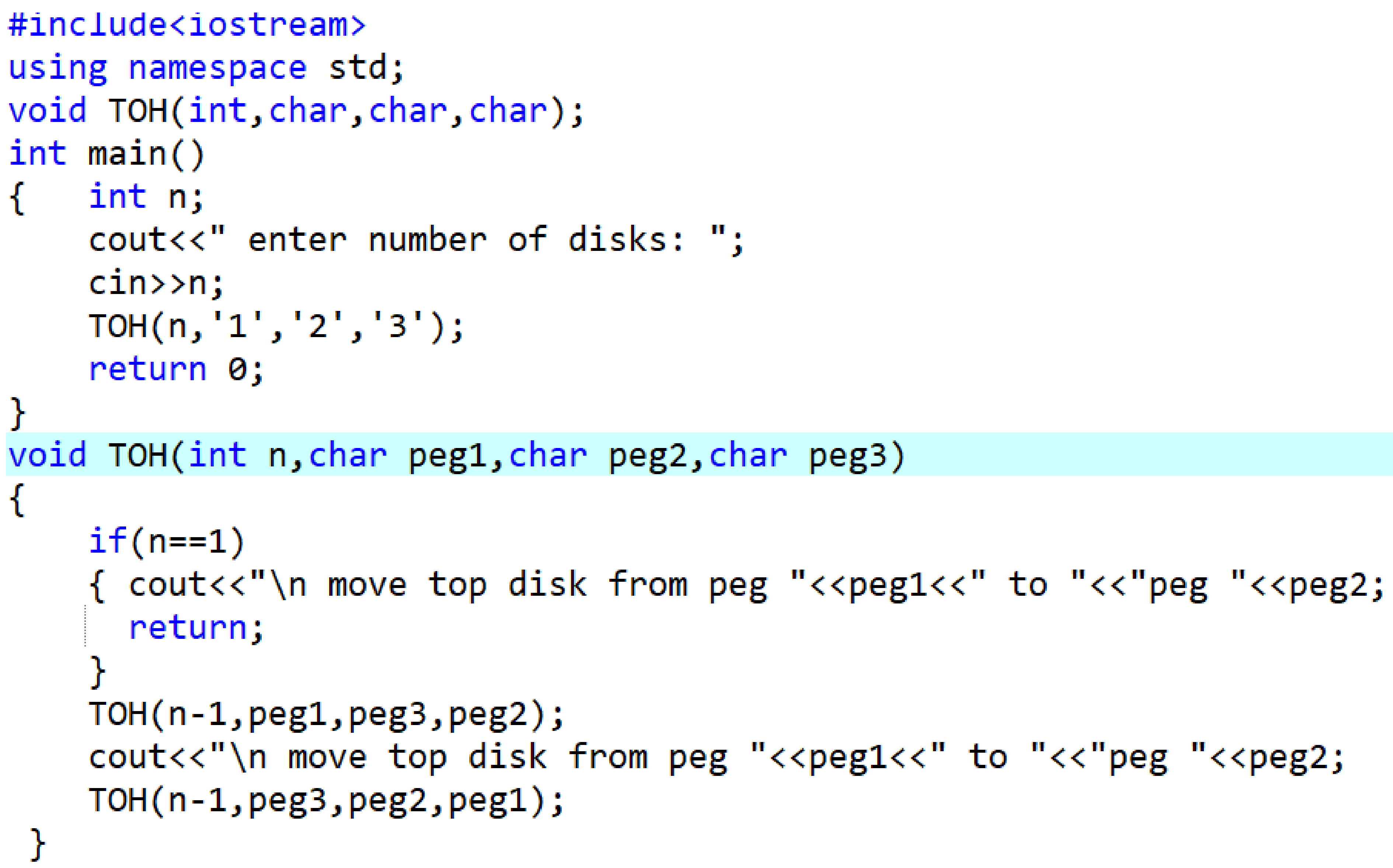
The rules of the puzzle

- allow to move one disk at a time from one peg to another
- always allow to place a smaller disk on the larger

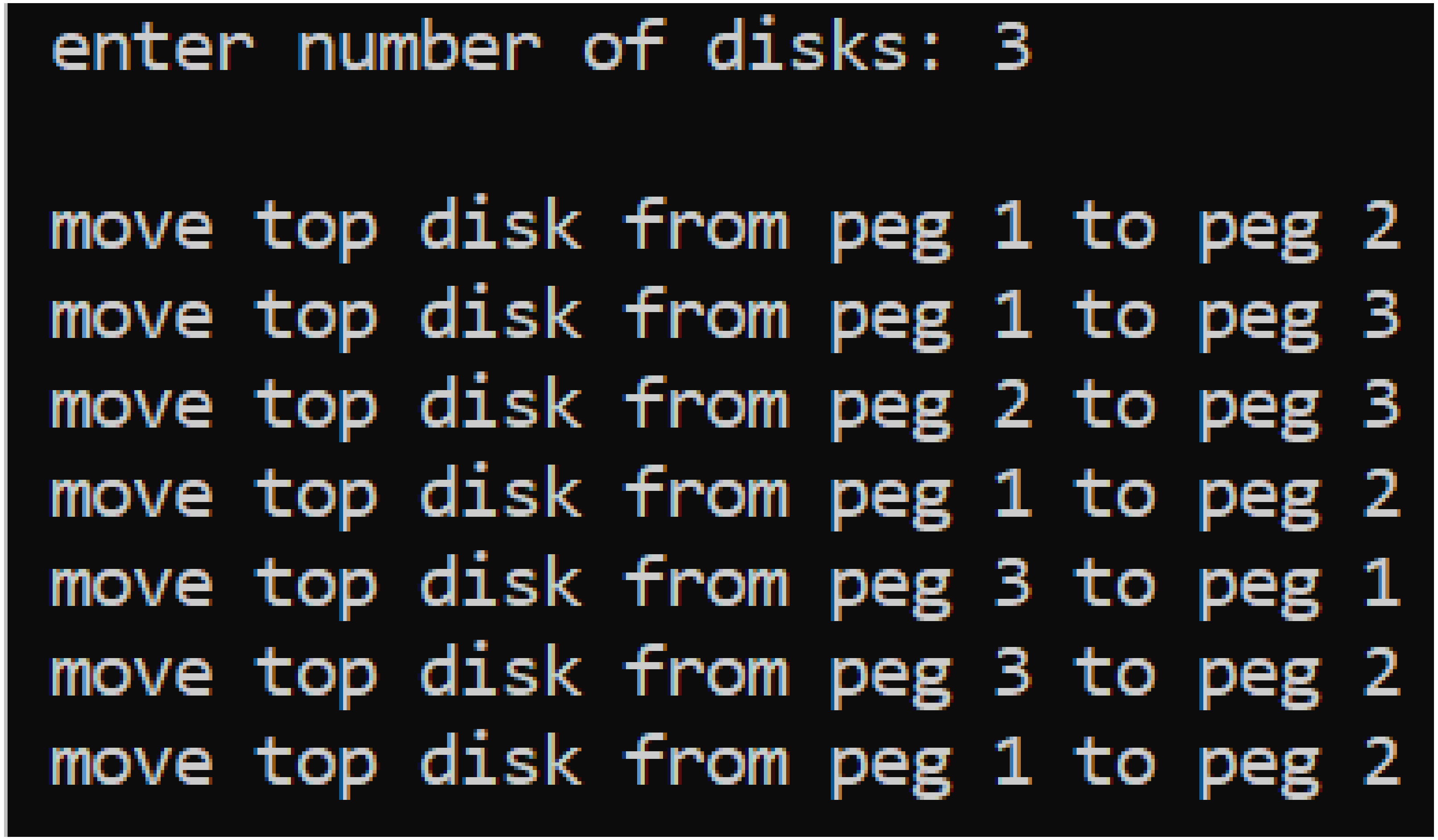
The goal of the puzzle is to have all the disks on the second peg in order of size, with the largest on the bottom

Algorithm

- Move top n-1 disks to peg-3 from peg-1
- Move bottom most nth disk to peg-2 from peg-1
- Move n-1 disks to peg-2 from peg-3

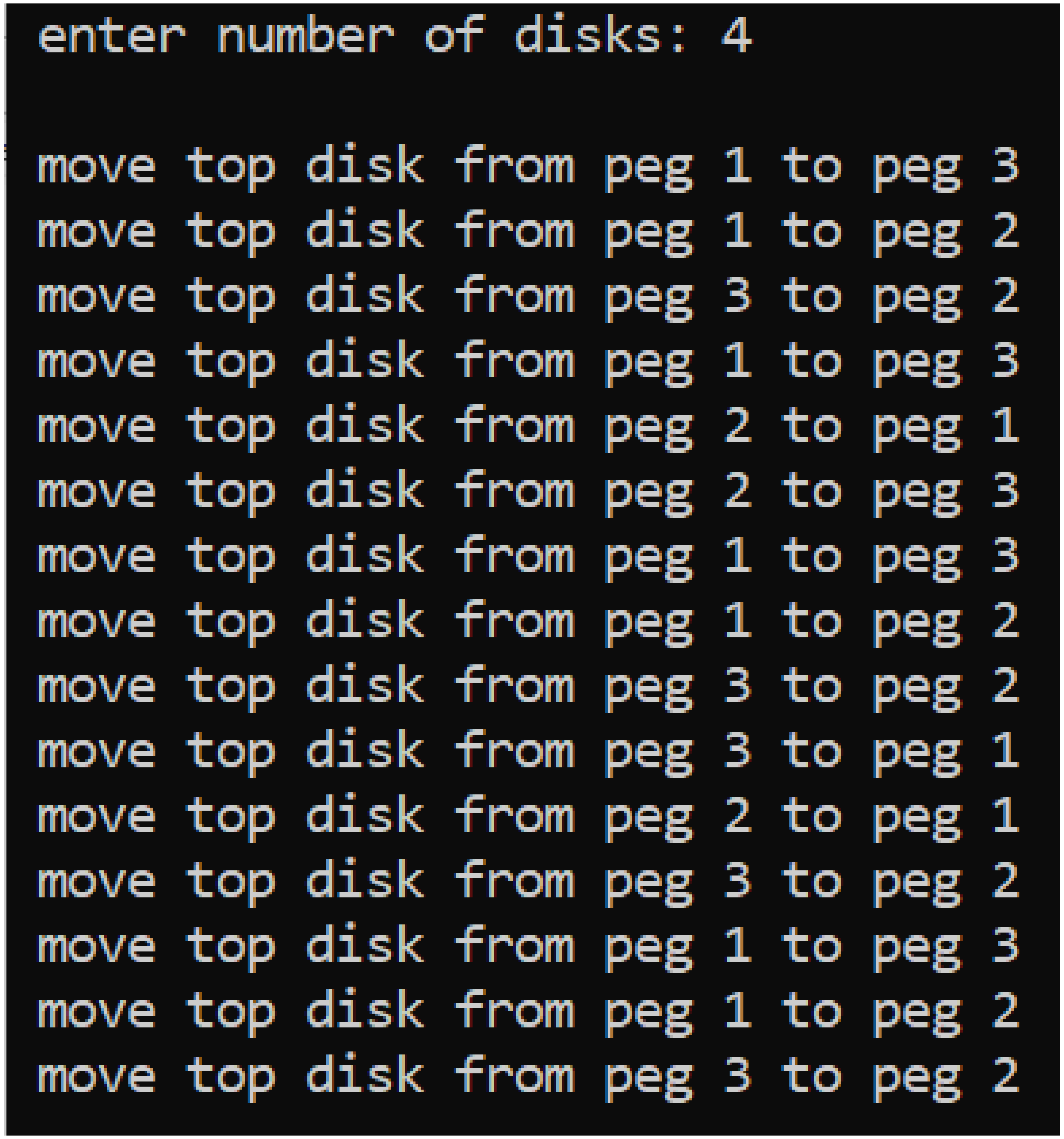


```
#include<iostream>
using namespace std;
void TOH(int,char,char,char);
int main()
{   int n;
    cout<<" enter number of disks: ";
    cin>>n;
    TOH(n,'1','2','3');
    return 0;
}
void TOH(int n,char peg1,char peg2,char peg3)
{
    if(n==1)
    { cout<<"\n move top disk from peg "<<peg1<<" to "<<"peg "<<peg2;
      return;
    }
    TOH(n-1,peg1,peg3,peg2);
    cout<<"\n move top disk from peg "<<peg1<<" to "<<"peg "<<peg2;
    TOH(n-1,peg3,peg2,peg1);
 }
```



```
enter number of disks: 3

move top disk from peg 1 to peg 2
move top disk from peg 1 to peg 3
move top disk from peg 2 to peg 3
move top disk from peg 1 to peg 2
move top disk from peg 3 to peg 1
move top disk from peg 3 to peg 2
move top disk from peg 1 to peg 2
```



```
enter number of disks: 4


move top disk from peg 1 to peg 3
move top disk from peg 1 to peg 2
move top disk from peg 3 to peg 2
move top disk from peg 1 to peg 3
move top disk from peg 2 to peg 1
move top disk from peg 2 to peg 3
move top disk from peg 1 to peg 3
move top disk from peg 1 to peg 2
move top disk from peg 3 to peg 2
move top disk from peg 3 to peg 1
move top disk from peg 2 to peg 1
move top disk from peg 3 to peg 2
move top disk from peg 1 to peg 3
move top disk from peg 1 to peg 2
move top disk from peg 3 to peg 2
```