GeeksMan
Recursion
Lesson 2

Power of Numbers

Given a number N, let the reverse of the number be R. The task is to print the output of the Expression pow(N,R), where pow function represents N raised to power R.

Note: As answers can be very large, print the result modulo 1000000007.

Input:

The first line of the input consists of an integer T denoting the number of test cases. Then T test cases follow. Each test case consists of a single line containing an integer N.

Output:

Corresponding to each test case, print in a new line, the output of the expression pow as described above.

Constraints:

1 <= T <= 103

1 <= N <= 1010

Example:

Input:

2

2

12

Output:

4

864354781

Explanation:

Testcase 1: The reverse of 2 is 2 and after raising power of 2 by 2 we get 4 which gives remainder as 4 by dividing 1000000007.

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;

long long int pw(long long int n,long long int p)
{
   if(p==0)
   return 1;
   long long int t;
   long long int m=1000000007;
   t=(pw(n,p/2));
  // cout<<t<<endl;
   if(p%2)
   {
       return (n*((t*t)%m))%m;
   }
   else
   return (t*t)%m;
}

long long int rev(long long int n)
{
   long long int r=0;
   while(n!=0)
   {
       r=(r*10)+n%10;
       n/=10;
   }
```

```
29.  // cout<<r<<endl;
30.   return r;
31.}
32.void solve()
33.{
34.long long int n;
35.cin>>n;
36.long long int p=rev(n);
37.cout<<pw(n,p);
38.
39.}
40.int main()
41.{
42.
43.   int t;
44.   cin>>t;
45.   while(t--)
46.   {
47.       solve();
48.       cout<<endl;
49.   }
50.}
```

Tower of Hanoi

The tower of Hanoi is a famous puzzle where we have three rods and N disks. The objective of the puzzle is to move the entire stack to another rod. You are given the number of discs N. Initially, these discs are in the rod 1. You need to print all the steps of disc movement so that all the discs reach the 3rd rod. Also, you need to find the total moves.

Note: The discs are arranged such that the top disc is numbered 1 and the bottom-most disc is numbered N. Also, all the discs have different sizes and a bigger disc cannot be put on the top of a smaller disc.

Input:

The first line of input is T denoting the number of test cases. T test cases follow. Each test case contains a single line of input containing N.

Output:

For each test case , print the steps and the total steps taken.

Constraints:

1 <= T <= 16

1 <= N <= 16

Example:

Input:

2

2

3

Output:

3

7

Solution :

```
#include <iostream>
#include<bits/stdc++.h>
using namespace std;
void toh(int s,int d,int h,int n)
{
   if(n==1)
   {
      cout<<"moving plate :"<<n<<"from "<<s<<"to "<<d<<endl;
      return;
   }
   toh(s,h,d,n-1);
   cout<<"moving plate :"<<n<<"from "<<s<<"to "<<d<<endl;
   toh(h,d,s,n-1);

}

int main() {
   int n;
   cin>>n;
   int s=1,h=2,d=3;
   toh(s,d,h,n);
   return 0;
}
```

Recursive sequence

A function f is defined as follows F(n)= (1) +(2*3) + (4*5*6) ... n. Given an integer n the task is to print the F(n)th term.

Input:

The first line of input contains an integer T denoting the number of test cases. Then T test cases follow. Each test contains an integer n.

Output:

For each test case print the nth term of the sequence. .

Constraints:

1 <= T <= 10

1 <= N <= 10

Example:

Input:

2

5

7

Output:

365527

6006997207

```
1. #include<iostream>
2. #include<bits/stdc++.h>
3. using namespace std;
4. long long int sum(int n)
5. {
```

```
6.      if(n==0)
7.    {
8.        return 1;
9.    }
10.   int cut=((n*(n+1))/2)+1;
11.   long long res=1;
12.   for(int i=cut;i<cut+n+1;i++)
13.   {
14.       res=res*i;
15.   }
16.   res+=sum(n-1);
17.   return res;
18.}
19.void solve()
20.{
21.   int n;
22.   cin>>n;
23.   int t=n-1;
24.   cout<<sum(t);
25.}
26.int main()
27.{
28.       int t;
29.   cin>>t;
30.   while(t--)
31.   {
32.       solve();
33.       cout<<endl;
34.   }
35.}
```

Number of paths

The problem is to count all the possible paths from top left to bottom right of a MxN matrix with the constraints that from each cell you can either move to right or down.

Example :

Input:
M = 3 and N = 3
Output: 6
Explanation:
Let the given input 3*3 matrix is filled
as such:
A B C
D E F
G H I
The possible paths which exists to reach
'I' from 'A' following above conditions
are as follows:ABCFI, ABEHI, ADGHI, ADEFI,
ADEHI, ABEFI
Expected Time Complexity: O(m + n - 1))

Expected Auxiliary Space: O(1)

Constraints:

$1 \leq M, N \leq 10$

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;
int no_of_path(int n,int m)
{
  if(n==1||m==1)
     return 1;
  return (no_of_path(n-1,m)+no_of_path(n,m-1));
}
void solve()
{
  int n,m;
  cin>>n>>m;
  cout<<no_of_path(n,m);
}
int main()
{
     int t;
  cin>>t;
  while(t--)
  {
     solve();
    cout<<endl;
  }
```