Recursion

29.03.2022

—

❖ What is Recursion?

Ans:- It is the method to call a function directly or indirectly. The function which is called itself is called a recursive function.

❖ Why recursion is needed?

Ans:- Recursion code is generally shorter and easier. It is most useful for the tasks that can be defined in terms of similar smaller tasks.

❖ Basic structure of a recursive function:

```
function f {
    if(test for base case)
        return some base case value;
    else if (test for another base case)
        return some other base case value;
    else
        return (some work and then recursive call);
}
```

❖ Example:

- Question 1:- Write a function to find factorial of n using recursion.

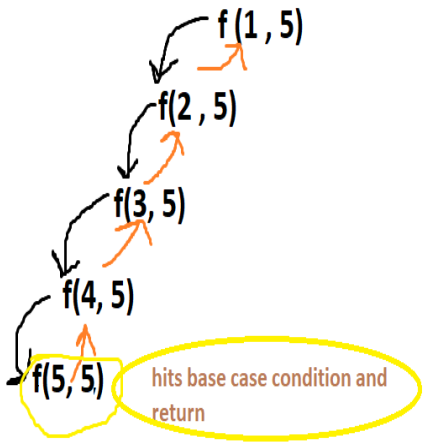
```
int fact(int num){
    if(n==1)          //--|
        return 1;     //   |
    else if(n==0)     //   |---Base case :
        return 1;     //--|fact 0 or 1 is 1
    else
        return num * fact(num-1);
}
```

Recursive call

Basic Recursion Problems

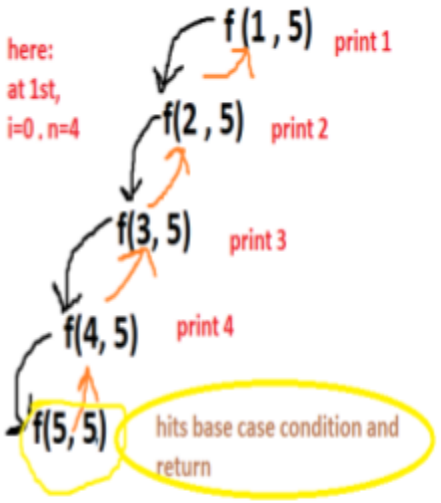
Q1. Print your name n times.

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

void print(int i, int n){
    if(i>n) //base case
        return;
    cout<<"Sneha"<<endl;
    f(i+1,n);
}

int main(){
    int n;
    cin>>n;
    print(1,n);
}
```



Q2, Print Linearly 1 to N.

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

void print(int i, int n){
    if(i>n)
        return;
    cout<<i<<endl;
    print(i+1,n);
}

int main(){
    int n;
    cin>>n;
    print(1,n);
}
```

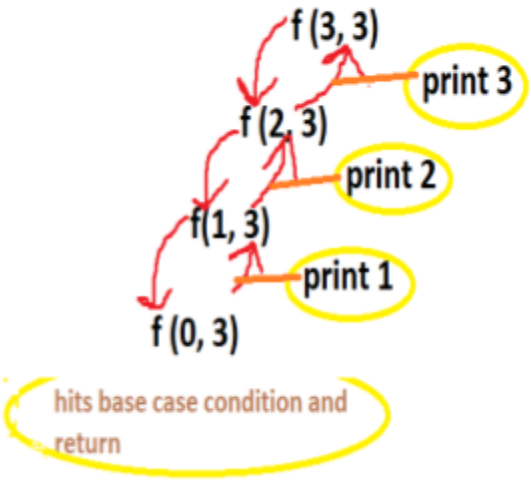


Q3. Print linearly in reverse order: N to 1

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

void print(int i, int n){
    if(i<1)
        return;
    cout<<i<<endl;
    print(i-1,n);
}

int main(){
    int n;
    cin>>n;
    print(1,n);
}
```

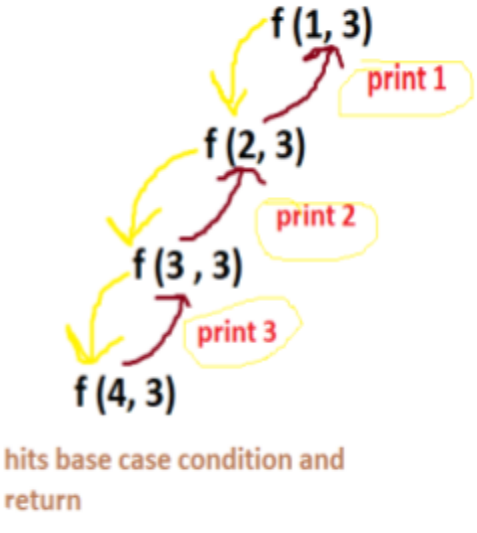
Q4. Print linearly 1 to N by backtracking.

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

void print(int i, int n){
    if(i<1)
        return;
    print(i-1,n);
    cout<<i<<endl;
}

int main(){
    print(3,3);
}
```



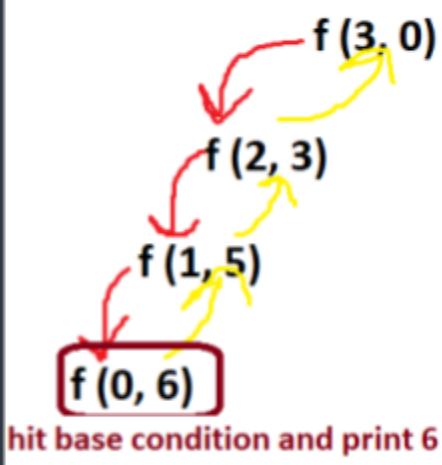
Q5. Print linearly N to 1 by backtracking.

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

void print(int i, int n){
    if(i>1)
        return;
    print(i+1,n);
    cout<<i<<endl;
}

int main(){
    print(1,3);
}
```

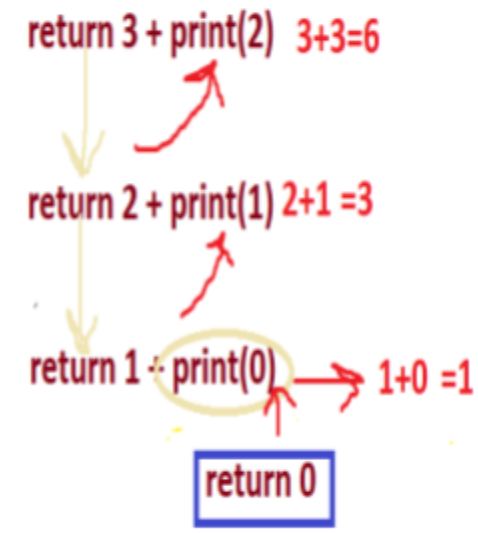


Q6. Print the sum of first n natural numbers: (i) using parameterized recursion. (ii) using functional recursion.

```cpp
//using parameterized recursion
#include<bits/stdc++.h>
using namespace std;

void print(int i, int n){
    if(i<1){
        cout<<sum;
        return;
    }
    print(i-1,sum+i);
}
int main(){
    print(3,0);
}
```

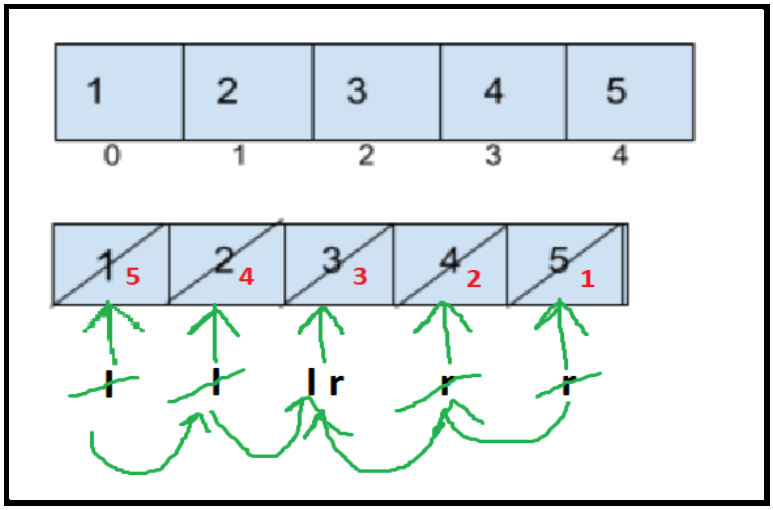



```cpp
//using functional recursion
#include<bits/stdc++.h>
using namespace std;

int print(int n){
    if(n==0)
        return 0;
    return n + print(n-1);
}
int main(){
    cout<<print(3);
}
```



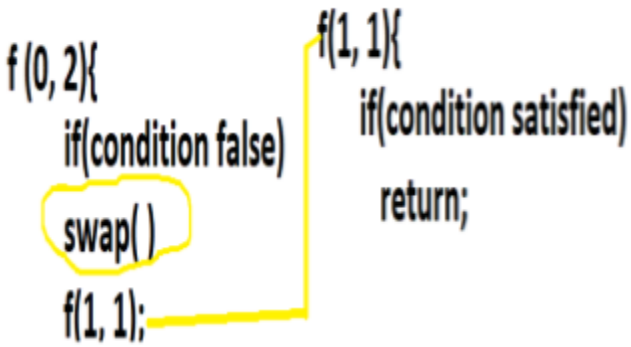
Problems of function recursion

Q1. Reverse array using recursion.



Method 1:

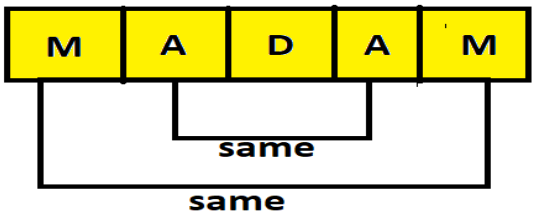
```
//Method 1
#include<bits/stdc++.h>
using namespace std;

int print(int l, int n, int arr[]){
    if(l>=r)
        return ;
    swap(arr[l], arr[n]);
    print(l+1, n-1, arr);
}
int main(){
    int arr[3]={1,2,3};
    print(0,2,arr);
}
```



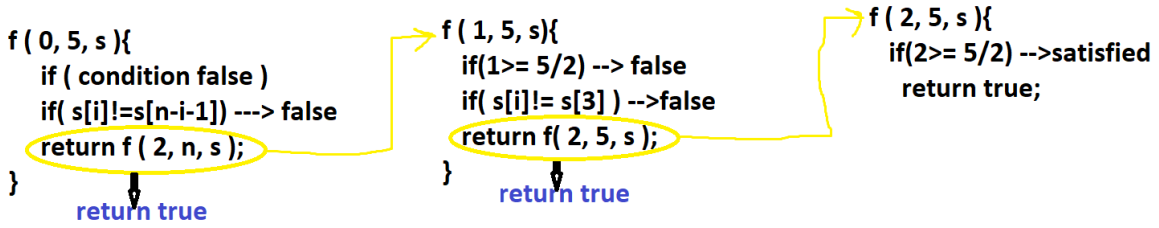
Method 2: Using single pointer

```
//Method 2
#include<bits/stdc++.h>
using namespace std;

void print(int i, int n, int arr[]){
    if(i>=n/2)
        return ;
    swap(arr[i], arr[n-i-1]);
    print(l+1, n);
}
int main(){
    int arr[3]={1,2,3};
    print(0,3);
}
```

Q2. Check if a string is a palindrome.

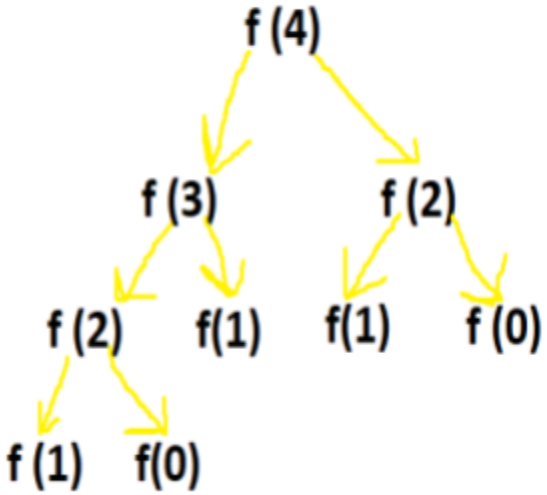



```
#include<bits/stdc++.h>
using namespace std;

bool print(int i, int n, string s){
    if(i>=n/2)
        return true;
    if(s[i]!=s[n-i-1])
        return false;
    return print(i+1, n, s);
}
int main(){
    string s= "MADAM";
    cout<<print(0, 5, s);
}
```



Q3. Find nth Fibonacci number using recursion.

```
clude<bits/stdc++.h>
ng namespace std;

fibo(int n){
if(n<=1)
    return n;
return fibo(n-1)+ fibo(n-2);


main(){
fibo(4);
```



Q3. Print all subsequence.

Input: arr = [3, 1, 2]

Output: [3], [1], [2], [3, 1], [1, 2], [3, 2], [3, 1, 2]

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

void print(int ind, vector<int> &ds, int arr[], int n){
    if(ind==n){
        for(auto it: ds)
            cout<<it<<" ";
        return;
    }
    ds.push_back(arr[ind]);
    print(ind+1, ds, arr, n);
    ds.pop_back();
    print(ind+1, ds, arr, n);
}

int main(){
    int arr[]={ 3, 1, 2};
    vector<int> vs;
    print(0,vs,arr, 3);
}
```

Q4. Printing all subsequences whose sum is k.

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

void printS(int ind, vector<int> &ds, int arr[],int sum, int s, int n){
    if(ind==n){
        if(s==sum){
            for(auto it: ds)
                cout<<ds<<" ";
            return;
        }
    }
    ds.push_back(arr[ind]);
    s+=arr[ind];
    printS(ind+1, ds, arr, sum, s, n);
    ds.pop_back();
    s-=arr[ind];
    printS(ind+1, ds, arr, sum, s, n);
}

int main(){
    int arr[]={1,2,1,3,4,0};
    vector<int> vs;
    printS(0,vs,arr,2,0,6);
}
```

Q5. Print one subsequence whose sum is k.

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

bool printS(int ind, vector<int> &ds, int arr[],int sum, int s, int n){
    if(ind==n){
        if(s==sum){
            for(auto it: ds)
                cout<<ds<<" ";
            return true;
        }
        return false;
    }
    ds.push_back(arr[ind]);
    s+=arr[ind];
    if(printS(ind+1, ds, arr, sum, s, n)==true);
        return true;
    ds.pop_back();
    s-=arr[ind];
    if(printS(ind+1, ds, arr, sum, s, n)==true);
        return true;
    return false;
}

int main(){
    int arr[]={1,2,1,3,4,0};
    vector<int> vs;
    printS(0,vs,arr,2,0,6);
}
```

Q6. Count the number of subsequences whose sum is k.

Basic Structure:

```
int f ( ){
    base case{
        return 1 --> condition satisfied.
        return 0 --> condition not satisfied.
        }
        l = f ( );
        r = f ( );
        return l+r;
    }
```

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

bool printS(int ind, int arr[],int sum, int s, int n){
    if(ind==n){
        if(s==sum)
            return 1;
        return 0;
    }
    s+=arr[ind];
    int l= printS(ind+1,arr, sum, s, n);
    s-=arr[ind];
    int r= printS(ind+1,arr, sum, s, n);
    return l+r;
}

int main(){
    int arr[]={1,2,1,3,4,0};
    printS(0,arr,2,0,6);
}
```

To be continued…