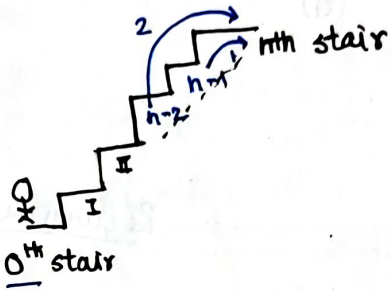
11th March

Week 7 Lec-2

Recursion Level-2

1. Climbing Stairs :-



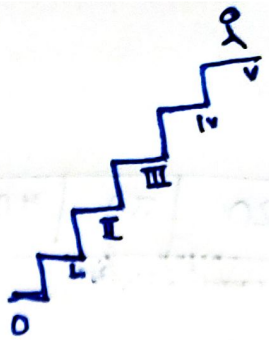
Steps allowed
- 1 stair at a time
- 2 stair at a time

Total ways to reach n^{th} stair $\Rightarrow$ $f(n) = f(n-1) + f(n-2)$

Find no. of ways to reach nth stair.

```
int climbStairs(int n){
    // base case
    if(n==0) return 1; // 0th stair पे पहुचने के तरीके
    if(n==1) return 1; // 1st stair पे पहुचने के तरीके
    int ans = climbStairs(n-1) + climbStairs(n-2);
    return ans;
}

if(n==1) return 1
if(n==2) return 2

int main(){
    int n;
    cin>>n;
    int ans = climbsStairs(n);
    cout<<ans<<endl;
    return 0;
}
```

$$f(5) = f(4) + f(3)$$



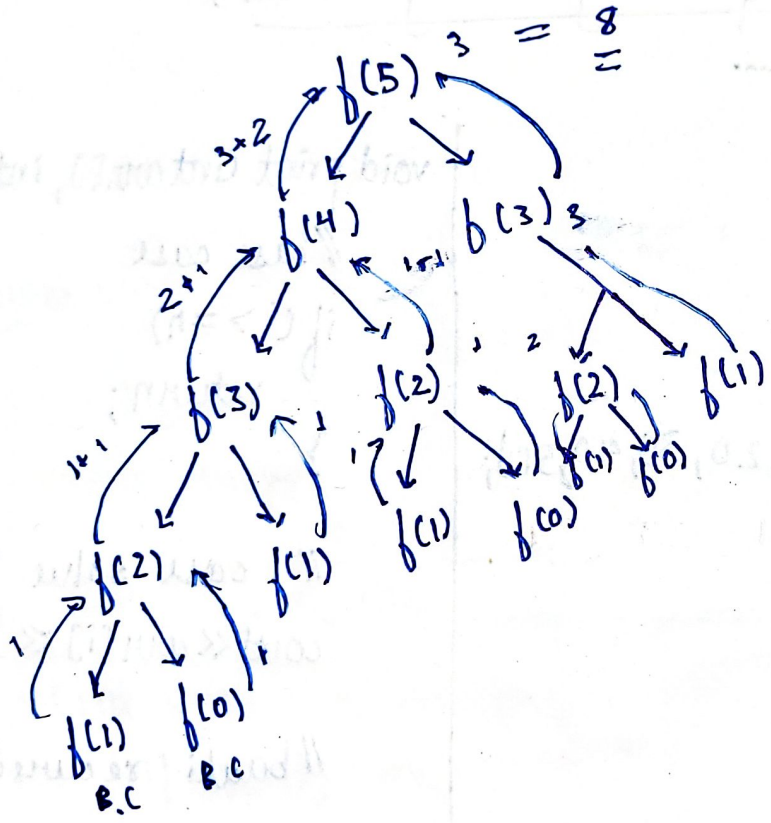
$$f(4) = f(3) + f(2)$$
$$= 3 + 2$$

$$f(3) = f(2) + f(1)$$
$$= 2 + 1$$

$$f(2) = f(1) + f(0)$$
$$= 1 + 1$$

$f(1) = 1$

$f(0) = 1$



Array

10	20	30	40	50

```
for(int i=0; i<n; i++){
        cout<< arr[i];
}
```

Loop

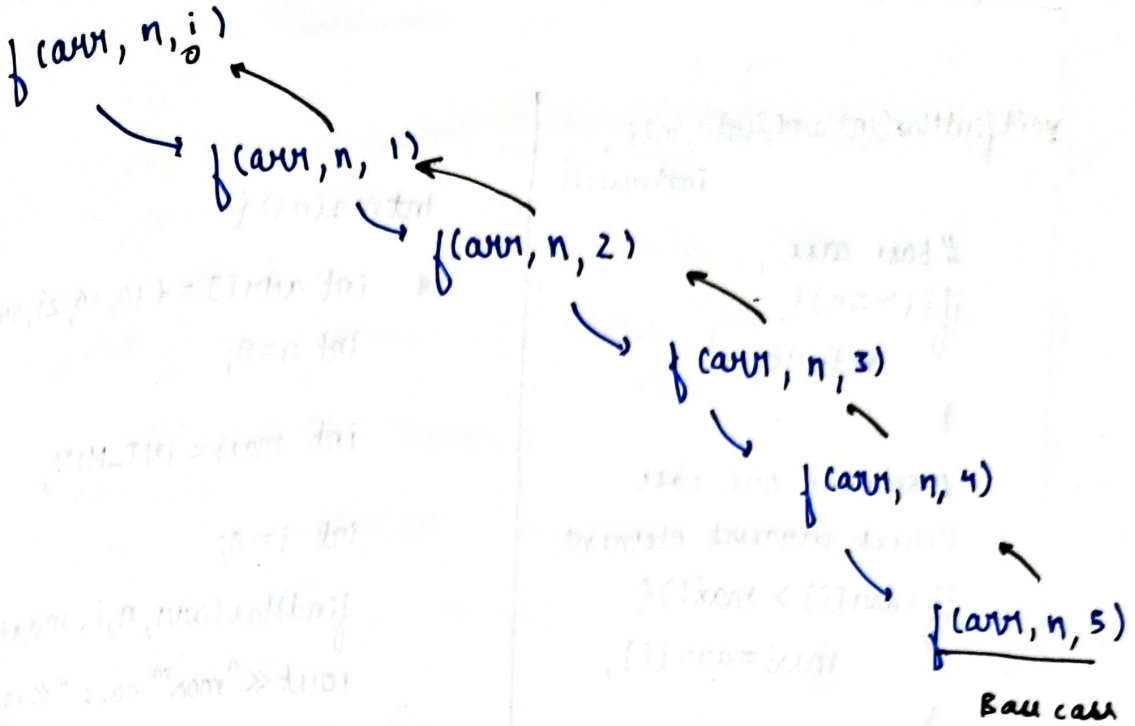
2. Using recursion printing Array elements :-

```
#include <iostream>
using namespace std;
int main(){
    int arr[5] = {10,20,30,40,50};
    int n=5;
    int i=0;
    print(arr, n, i);
    return 0;
}
```

```
void print(int arr[], int n, int i){
    // base case
    if(i>=n)
        return;
    }
    // 1 case solve krdia
    cout<< arr[i] << " ";
    // baaki recursion karlega
    print(arr, n, i+1);
}
```

f(arr, n, i)
p=0
print arr[0]
f(arr, n, i+1)
print arr[1]
f(arr, n, i+1) i+1=2
print arr[2]

10	20	30	40	50
0	1	2	3	4



3. Maximum

Ml:- Using loops

```
int maxi = INT_MIN
for(int i=0; i<n; i++){
    if(arr[i] > maxi)
        maxi = arr[i];
}
```

OR

```
for(int i=0; i<n; i++){
    maxi = max(maxi, arr[i]
}
```

M2- Recursion

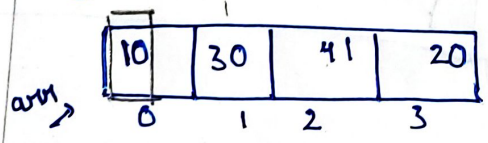
```
void findMax(int arr[], int n, int i,
                        int &maxi){

    // base case
    if(i>=n){
        return;
    }

    // solving one case
    //check current element
    if(arr[i] > maxi){
        maxi = arr[i];
    }
    // baake recursion smhalega
    findMax(arr, n, i+1, maxi);
}
```

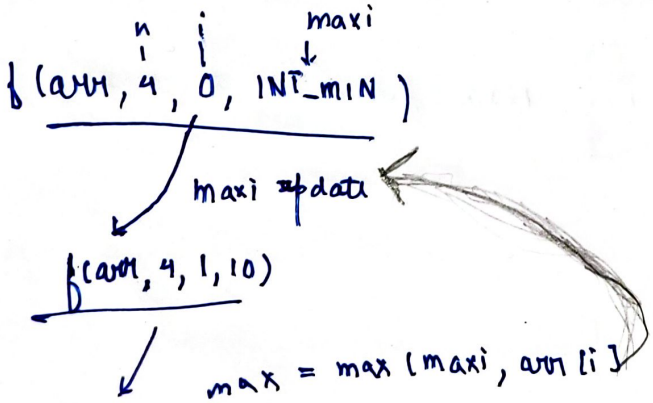
```
int main(){
    int arr[] = {10,30,21,44,32,17,13,66};
    int n=8;

    int maxi = INT_MIN;

    int i=0;
    findMax(arr, n, i, maxi);
    cout << "max no.: " << maxi << endl;

    return 0;
}
```

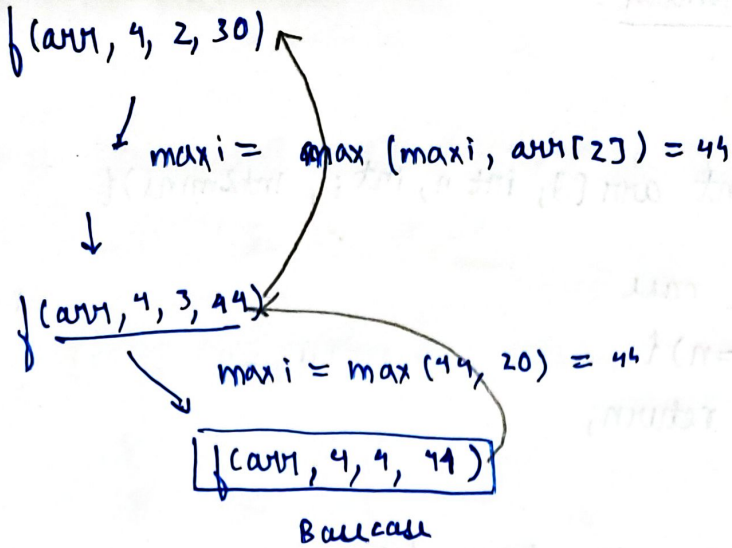
O/P - 66

Note- Don't use i++ in this. (less confusion)

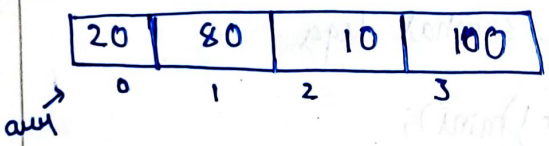
Dry run :-







Dry Run



f(arr, 4, 0, maxi)

maxi update

f(arr, 4, 1, 20) ⟶ main() — mai pahuch k INT_MIN print hojaega

max = max (maxi, arr[i])
20 80 1
=80

f(arr, 4, 2, 80)

maxi = (max (maxi, arr[2]))
80 , 10

f(arr, 4, 3, 80)

maxi = (max (maxi, arr[3]))
80 , 100

f(arr, 4, 4, 100)

Base case

4. To find Minimum :-

```
void findMin(int arr[], int n, int i, int&mini){

    //base case
    if(i>=n){
        return;
    }

    //1 case solve krna pdega
    mini = min(mini, arr[i]);

    //baaki recursion sambhal lega
    findMin(arr, n, i+1,mini);

}

int main(){
    int arr[] = {10, 30, 50, 15, 12, 60};
    int n = 6;
    int mini = INT_MAX;
    int i=0;
    findMin(arr,n,i,mini);
    cout<<mini<<endl;

    return 0;
}
```

Op:

10

3. To Find Maximum

```
void findMax(int arr[], int n, int i, int &maxi){
    //base case
    if(i>=n){
        return;
    }
    // 1 case solve krna hai
    maxi = max(maxi, arr[i]);

    //baaki recursion smhal lega
    findMax(arr, n, i+1, maxi);
}
int main(){
    int arr[] = {10, 20, 90, 70, 20};
    int n = 5;
    int maxi = INT_MIN;
    int i=0;
    findMax(arr, n, i, maxi);
    cout<< maxi << endl;
    return 0;
}
```

O/P

90

5. To check if in i/p ~~एक~~ string hai

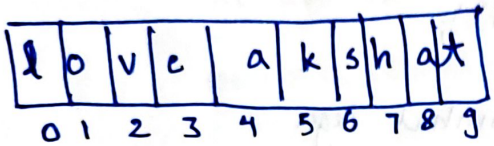
string str = "love akshat"

To find: Key → 't'

↳ t is present in str or not

(using Recursion)

Ans:




```
bool checkKey(string &str, int i, int &n, char &key) {
    //base case
    if (i >= n) {
        //key not found
        return false;
    }
    //I can solve kr do
    if (str[i] == key)
        return true;

    // baaki recursion samhal lega
    return checkKey(str, i+1, n, key);
}
```

Note- By reference send krne se space complexity कम lagegi.
TLE भी हट सकता है!

```
int main(){
    string str = "loveakshat";
    int n = str.length();
    char key = 't';
    int i = 0;
    bool ans = checkKey(str, i, n, key)
    cout<< ans <<endl;
    return 0;
}
```

O/P
1

Now, To return index of key if found, else -1.

→ Instead of bool, use int

→ Use -1, where false is returned.

and 'return i' instead of return true.

In main()

→ Nothing would be changed.

Making it more correct:-

अगर akshat — a is coming 2 times तो
print both loc^n

eg:

```
Found at : 0
Found at : 4
```

Code:

```
void checkKey(string& str, int i, int& n, char& key){
    //base case
    if(i>=n){
        //key not found
        return;
    }
    //1 case solve krdo
    if(str[i] == key){
        cout<<"Found at:"<<i<<endl;
    }
    //baaki recur^n
    return checkKey(str, i+1, n, key);
}
```

```
int main(){
    string str = " lovu akshat ";
    int n = str.length();
    char key = 'a';
    int i=0;
    checkKey(str, i, n, key);

    return 0;
}
```

6. **Problem statement**

i/p → 647

o/p → print all digit of this no.

```
Void printDigits(int n){
        // base case
        if(n==0){
            return;
        }
        // 1 case mei solve krunga
        int digit = n%10;
        cout<<digit<<" ";
        int newValueofN = n/10;
        // baaki recursion smhalega
        printDigits(newValueofN);
}

int main(){
    int n=647;
    printDigits(n);

    return 0;
}
```

Tail Recursion

o/p

7 4 6

For seedha print

Do, head recursion :-

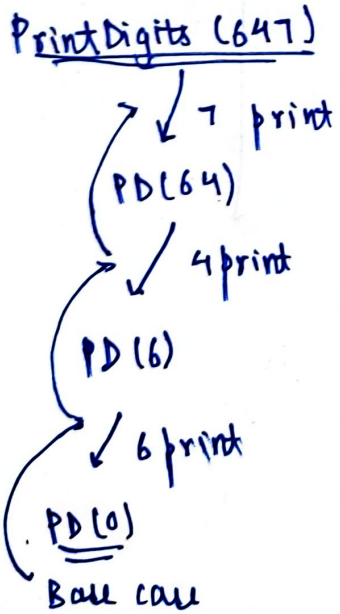
```
if (n==0){
    return;
}
int newValueOfN = n/10;
// baaki recurn
printDigits(newValueOfN);
// 1 case mai solve krungg
int digit = n%10;
cout << digit << " ";
```

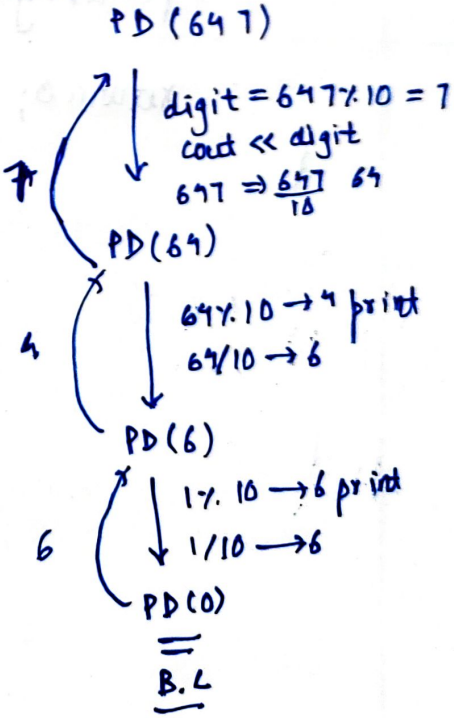
O/P

~~6 4 7~~ 6 4 7

~~4 6 7~~

Dry Run





HW Revise — 5 eg. pe Dry Run (Video ek baar aur dekho)

will fail for '0'

```
int main(){
    int n=0;
    printDigits(n);
    return 0;
}
```

O/P

aaega hi nhi.

To handle this :-

```
int main(){
    int n=2000;
    if(n==0)
        cout<<0<<endl;
    printDigits(n);
    return 0;
}
```

O/P

2000