Recursion on Arrays

→ faith: prints array ele arr[0] .... arr[n-1]

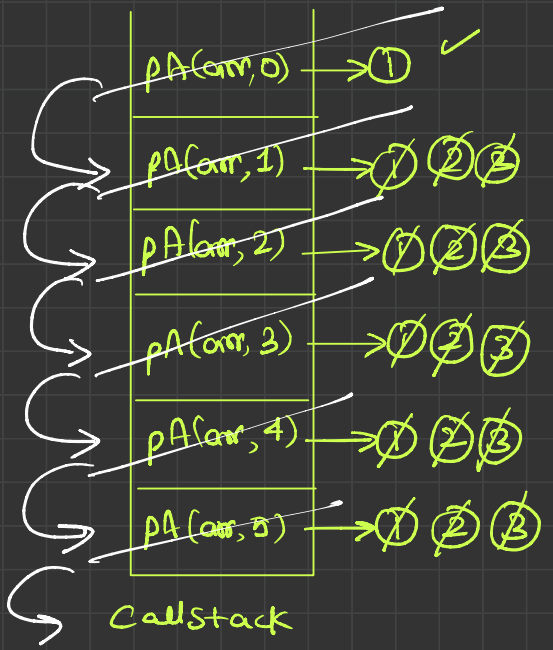
```
void printArray(int[] arr, int n)
{
    if(n==0)
        return;

    printArray(arr, n-1);   → print array ele arr[0] ... arr[n-2]
    System.out.print(arr[n-1]+" ");
}
```

```
void printArray(int[] arr, int n)
{
    ① if(n==0)
        return;
    ② printArray(arr, n-1);
    ③ System.out.print(arr[n-1]+" ");
}
```

```
int[] arr = {10, 12, -1, 5, 6}
int n = 5
```

o/p

10 12 -1 5 6 ✓



Q arr[] = {10, 4, 5, -1} o/p -1 5 4 10

→ faith: print array ele from arr[n-1] ..... arr[0]

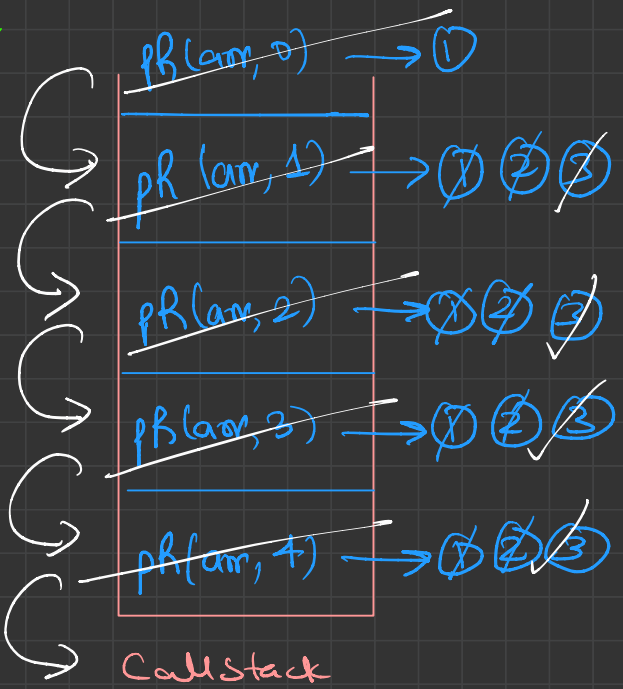
```
void printReverse(int[] arr, int n)
{
    if (n == 0) return;
    System.out.print(arr[n-1] + " ");
    printReverse(arr, n-1);   ⇝ arr[n-2] ..... arr[0]
}
```

```
void printReverse(int[] arr, int n)
{
    ① if (n == 0) return;
    ② System.out.print(arr[n-1] + " ");
    ③ printReverse(arr, n-1);
}
```

arr[] = { 10, 4, 5, -1 } (0, 1, 2, 3)

N = 4

o/p: -1 5 4 10



get Smallest

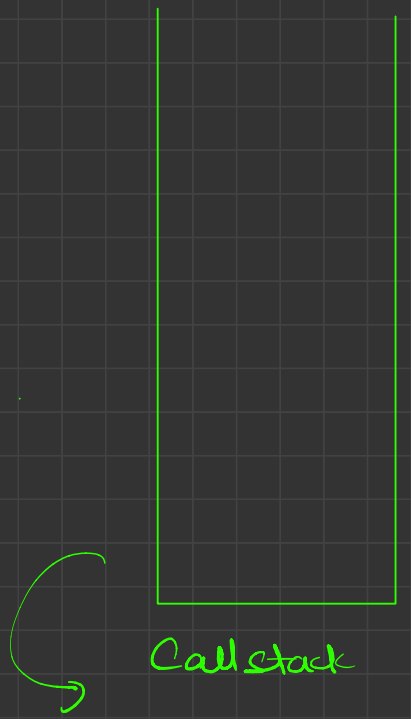
arr[] = {2, 5, -1, 100, 1, 20} idx = 0

faith: returns min ele present in this array starting from idx to arr.length - 1.

```
int getMin(int[] arr, int idx)
{
    if(idx == arr.length) return Integer.MAX_VALUE;

    int a = getMin(arr, idx+1);

    return Math.min(arr[idx], a);

}
```

arr[] = { 2, 5, -1, 100 } ✓ idx len = 0

```
int getMin(int[] arr, int idx)
{
    (1) if(idx == arr.length) return Integer.MAX_VALUE;
    (2) int a = getMin(arr, idx+1);
    (3) return Math.min(arr[idx], a);
}
```



Palindrome

arr[] = { 1, 2, 3, 2, 1 }

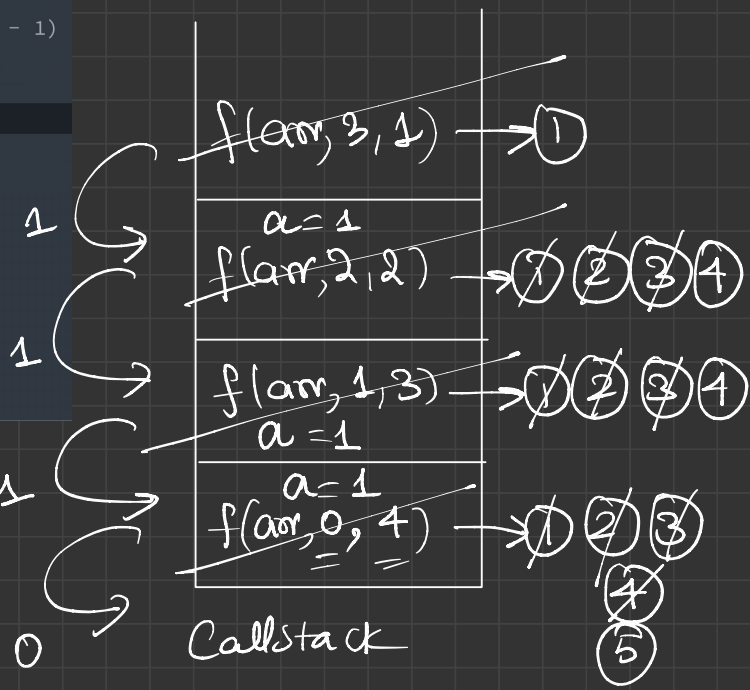
0 1 2 3 4

→ return is array between index begin and start is palindromic or not.

```
int isPalindromic(int[] arr, int begin, int end)
{
    if (begin > end)
        return 1;

    int a = isPalindromic(arr, begin+1, end-1);
    if (a == 0) return 0;
    else
        if (arr[begin] == arr[end]) return 1;
        else return 0;
}
```

```java
// Faith: returns if array is palandromic between index begin and end
public static int isPalindromic(int[] arr, int begin, int end) {
    if (begin > end) {
        return 1;
    }

    // returns if array is palandromic between index (begin + 1) and (end - 1)
    int a = isPalindromic(arr, begin + 1, end - 1);

    if (a == 0) {
        return 0;
    } else {
        if (arr[begin] == arr[end]) {
            return 1;
        } else {
            return 0;
        }
    }
}
```

arr[] = {1, 2, 3, 2, 4}



first Occurance

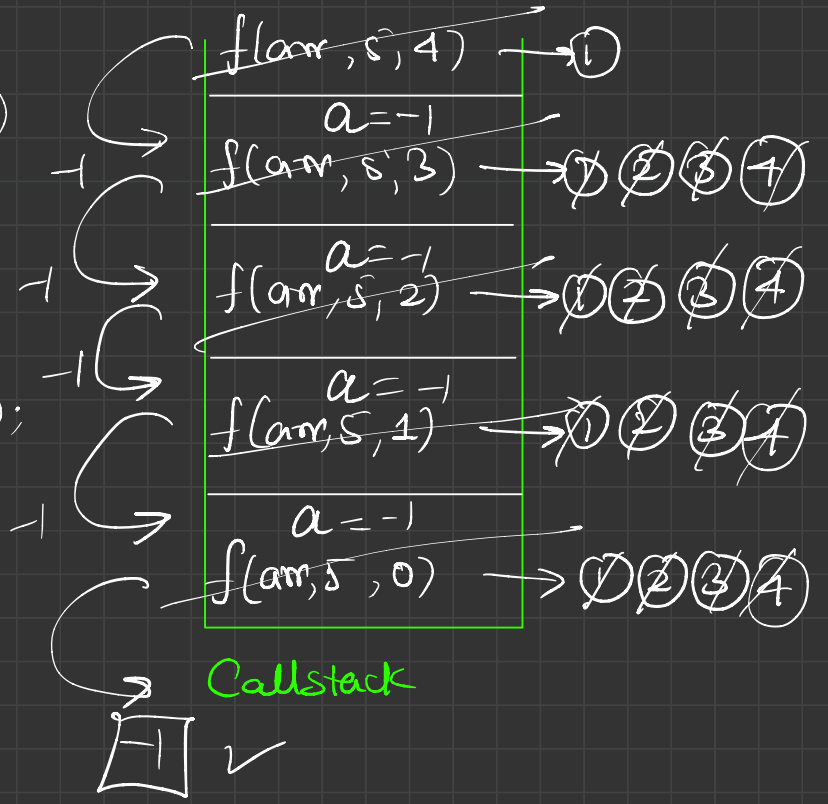
```
                    0  1  2  3  4  5  6  7
int[] arr = { 1, 2, 3, 2, 2, 4, 5, 6 }      T = 2      startIndex = 0
```

→ faith: first Occ. of the Element in the Array starting from startIndex

```
int firstIndex(int[] arr, int T, int startIndex)
{
    if (startIndex == arr.length) return -1;

    if (arr[startIndex] == T)
        return startIndex;

    int a = firstIndex(arr, T, startIndex + 1);
    return a;
}
```

int[] arr = { 1, 2, 3, 2 } (0 1 2 3) T = 5 startIndex = 0

```
int firstIndex(int[] arr, int T, int startIndex)
{
  (1) if (startIndex == arr.length) return -1;
  (2) if (arr[startIndex] == T)
          return startIndex;
  (3) int a = firstIndex(arr, T, startIndex+1);
  (4) return a;
}
```



Last Occ

→ faith; returns last index of ele T in array arr, ending with ei.

```
int LastIndex (int[]arr, int T, int ei)
{   if (ei < 0) return -1;

    if(arr[ei] == T)
        return ei;

int a = LastIndex(arr, T, ei-1);
    return a;
}
```

arr[] = { 0:1, 1:2, 2:3, 3:2, 4:2, 5:3, 6:4 }

T = 2

```java
// Faith: returns last occ of T in the array starting from startIndex
static int lastIndex(int A[], int T, int startIndex) {
    if (startIndex == A.length) {
        return -1;
    }

    int a = lastIndex(A, T, startIndex + 1);
    if (a != -1) {
        return a;
    } else {
        if (A[startIndex] == T) {
            return startIndex;
        } else {
            return -1;
        }
    }
}
```

1
2
3
4
5

find all index

→ print all index of x in arr from 0 → N-1

```
void findIndex(int[]arr, int N, int x)
{
    if(N==0)
        return;

    findIndex(arr, N-1, x);   ~> [0 -> N-2]

    if(arr[N-1] == x)
        print(N-1);
}
```

N^{th} Even fibonacci

1st	2nd	3rd	4th	5th	6th	7th	8th	9th
1	1	2	3	5	8	13	21	34	55	89	144

$$f(n) = f(n-1) + f(n-2)$$

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

$$= f(n-2) + 2 \times f(n-3) + f(n-4)$$

$$= f(n-3) + f(n-4) + 2 \times f(n-3) + f(n-4)$$

$$= 3 \times f(n-3) + f(n-4) + f(n-5) + f(n-6)$$

$$= 3 \times f(n-3) + f(n-3) + f(n-6)$$

$$\boxed{f(n) = 4 \times f(n-3) + f(n-6)}$$

Even Fibonacci Series

$$EF(n) = 4 * EF(n-1) + EF(n-2)$$

$n = 1 \longrightarrow 2$

$n = 2 \longrightarrow 8$