Page No.:
Date : / /

har recursive code internally iterative hi chlta hai

ruler pattern

```
void pat(int n)
{   if(n==0) return;
    pat(n-1);
    for(int i=1;i<=n;i++) SOP("-");
    SOPln();
    pat(n-1);
}
```

English ruler

```
void er(int major, int len)
{   for(int i=0; i<len; i++)
    {   for(int cnt=1; cnt<=major; cnt++) SOP("-");
        SOPln(i);
        pat(major-1)
    }
    for(int cnt=1; cnt<=major; cnt++) SOP("-");
    SOPln(len);
}
```

Array Recursion

BP : (0, arr) SP : (1, arr)

```
void print(arr[], idx)
{   if(idx == arr.length) return
    SOPln(arr[idx])
    print(arr, idx+1)
}
```

dry run

print(arr, 0) → 10 → print(arr, 1) → 20 → print(arr, 2) → 30 → print(arr, 3) → 40 → print(arr, 4)

Page No.:
Date : / /

Array reverse by reverse

```
if (idx == arr.length) return
reverse (arr, idx+1)
SOPln (arr[idx]);
```

Max of array

```
max (int arr[], int idx)
{
    if (idx == arr.length - 1) return arr[idx];
    int sp = max (arr, idx+1)
    return Math.max (arr[idx], sp)
}
```

Min of array

```
min (int arr[], int idx)
{
    if (idx == arr.length-1) return arr[idx]
    int sp = min (arr, idx+1)
    return Math.min (arr[idx], sp)
}
```

first Occurance

```
firstocc (int arr[], int idx, int key)
{
    if (idx == arr.length) return -1;
    if (arr[idx] == key) return idx
    else
        return firstocc (arr, idx+1, key)
}
```

Page No.:
Date: / /

last occurance

BP = (idx, key) SP = (idx+1, key)

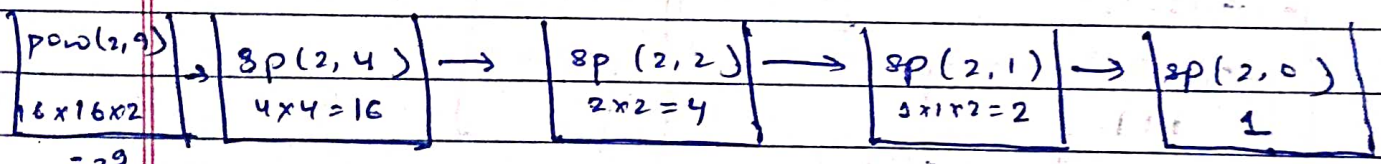
```
lastOcc(arr, idx, key)
{
    if(idx == arr.length) return -1
    int sp = lastOcc(idx+1, arr, key)
    if(sp == -1 && arr[idx] == key) return idx
    return sp
}
```

a^b

BP = pow(a, b) SP = pow(a, b/2)

```
{
  if(b == 0)        return 1
  int sp = pow(a, b/2)
  if(b%2 == 0)      return sp*sp
  else              return sp*sp*a
}
```

dry run pow(2, 9)



factorial