Data Segment

static var -DS

Day-1

PAGE No.

DATE 1/10/21

* Recursion -

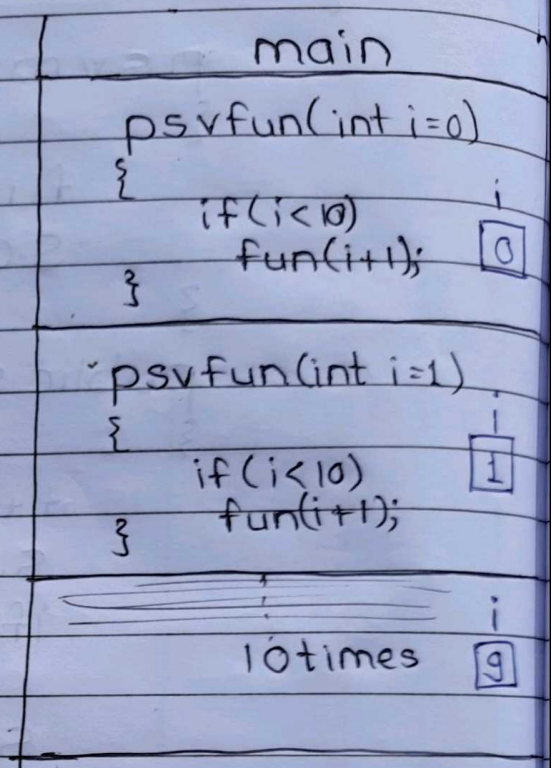
```
public class Main
{
    psvm(String [] args)
    {
        fun(0);
    }
    public static void fun(int i)
    {
        SOP(i+", ");
        if(i<10)
            fun(i+1);
    }
}
```

Local var

Stack वर प्रत्येक function call ला नवीन जागा मिळते.

Direct recursion

stack ↓

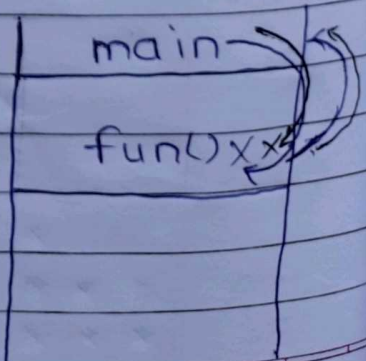


Here i=10 X (Removed from stack)
∴ Recursion stop
then it goes upward i=9 then i=8 ... i=2 all removed from stack i=1 also removed from stack
In FILO all are removed (9 to 1) & main removed program stop.

Recursion imp points

1) Function मध्ये कमीत कमी 1 var पाहिजे. (Local var)
2) fun ने fun ला call केल पाहिजे. (स्वतःला)
3) Recursion मधून बाहेर यायला condition पाहिजे. (exit)

```
public class Main
{
    psvm(String [] args)
    {
        for(int i=0; i<10; i++)
            fun(0);
    }
}
```



Here fun is not taking different place in stack so it is not recursion

एक function ला stack वर 1 पेक्षा जास्त वेळा जागा मिळते तेव्हा Recursion occur होत.

PAGE No.
DATE / /

```
public static void ~~main~~ fun(int i)
{
    SOP(i + ", ");
}
}

public class Main{
    psvm(String[] args)
    {
        fun(0);
    }
    public static void fun(int i)
    {
        show();
    }
    private void show()
    {
        main();
    }
}
```

main
fun
show
main
fun
show

(Indirect recursion)

Direct recursion - function calls to itself is direct recursion.

```
public class Main{
    psvm(String [] args)
    {
        fun(0);
    }
    public static void fun(int i)
    {
        S.O.P(i + ", ");
        if(i<10)
            fun(i+1);
    }
}
```

Here every fun call printing only 1 value

// 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

PAGE No.
DATE / /

```
public static void fun(int i)
{
    S.OP (i, " , "); // 0,1,2,3,4,5,6,7,8,9,10
    if (i<10)
    - fun(i+1);
    else
        SOP ();
    SOP(i, ", "); // 10,9,8,7,6,5,4,3,2,1,0
}
```

(Forward) F.R (जातोना)

RR (Reverse) (येतोना)

```
public class Main{
    p s v m(String [] args)
    {
        printStar(1);
    }
    private static void printStar(int i)
    {
        for(int j=0; j<i; j++)
        {
            SOP("*");
        }
        SOPln();
        printStar(i+1);
    }
    {
        for(int j=0; j<i; j++)
        {
            SOP("*");
        }
        SOPln();
        if(i<3)
            printStar(i+1);
    }
```

infinite

```
*
**
***
```

```
*
* *
* * *
* * *
* *
*
```

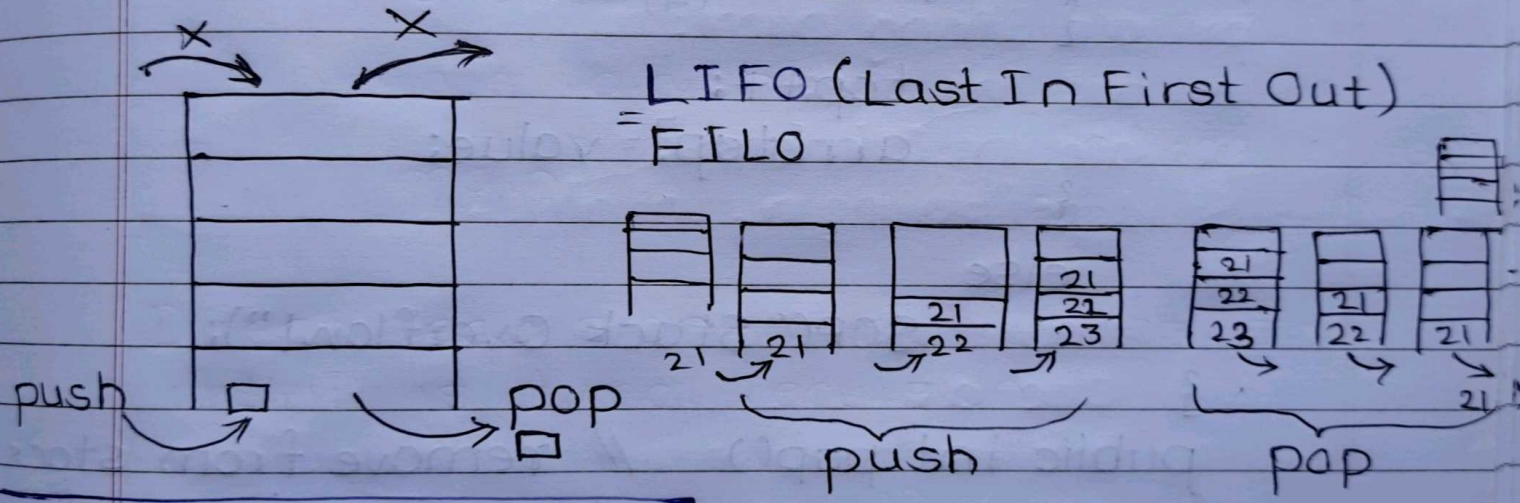
```
for(int j=0; j<i ;j++)
{
    SOP("*");
}
SOPln( );
if (i<3)
    printStar(i+1);
for(int j=0; j<i; j++)
{
    SOP("*");
}
SOPln();
```

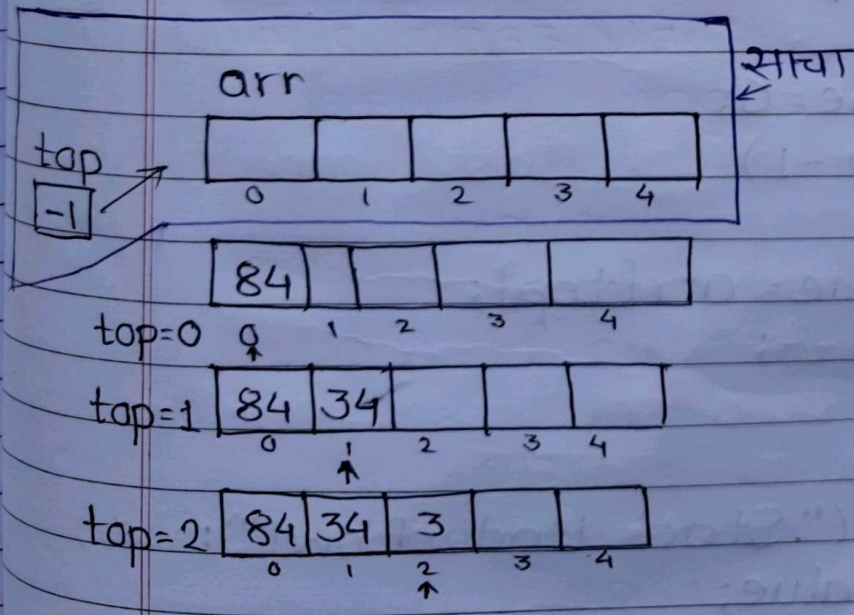
Forward

Reverse

(Stack)

* Array- (Classes & Logic)





push(23) / pop(23)

```
int [] arr; (Container)
int top;
```

* Data Structure *

Day-1

PAGE No.

DATE / /

* Stack - What is stack?

```
public class Stack
{
    int [] arr;
    int top;
    public Stack()
    {
        top=-1;
        arr=new int[5];
    }
    public void push(int value) // insert on stack
    {
        if(top< arr.length-1)
        {
            top++;
            arr[top]= value;
        }
        else
            SOP("Stack Overflow!");
    }
    public int pop()  // remove from stack
    {
        int value=0;
        if(top>-1)
        {
            value= arr[top];
            top--;
        }
        else
            SOP("Stack Underflow!");
        return value;
    }
```

index length (4)

Debug -
Breakpoint - stop on specific condition

PAGE No.
DATE / /

```
public String toString() // printing
{
    String str = " ";
    if(top == -1)
    {
        str = "Stack is Empty!";
    }
    else
    {
        for(int j=top; j>=0; j--)
        {
            str = str + arr[j];
        }
        str = str + "\n";
    }
    return str;
}
}
public class Main
{
    psvm(String [] args)
    {
        Stack s1 = new Stack();
        s1.push(34);
        s1.push(84);
        s1.push(15);
        s1.pop();
        s1.push(16);
        s1.pop();
        s1.push(10);
        SOP(s1);
    }
}
//10,84,34,
```

```
public Stack (int size)
{
    top = -1;
    arr = new int (size);
}
```

User dependent size

```
In main —
 Stack s2 = new Stack(25);
```

* Student Stack -

```
// Student
public class Student{
    private static int classNumber;
    private String name;
    private int age;
    private int [] marks;
    Student(String n, int a)
    {
        name = n;
        age = a;
        marks = new int[1];
        marks[0] = 0;
    }
    public static int getClassNumber()
    {
        return classNumber;
    }
    public static void setClassNumber(int classNu
    {
        Student.classNumber = classNumber;
    }
    public static void printClassNumber()
    {
```

```
        SOP("ClassNumber: "+classNumber);
    }
    Student(String n, int a, int [] m)
    {
        name=n;
        age=a;
        marks=m;
    }
    public void setMarks(int [] m)
    {
        marks=m;
    }
    public int [] getMarks()
    {
        return marks;
    }
    public String toString()
    {
        String str="Name:"+name+", age:"+
            age+"., Marks:"+getMarksInString()
            +"\n";
        return str;
    }
    private String getMarksInString()
    {
        String str="{";
        for(int i=0; i<marks.length; i++)
        {
            str=str+marks[i]+",";
        }
        str=str+"}";
        return str;
    }
}
```

```
// Stack
public class Stack()
{
    int Student [] arr;
        int top;
    public Stack()
    {
        top = -1;
        arr = new Student [5];
    }
    public void push (Student value)
    {
        if(top < arr.length-1)
        {
            top++;
            arr [top] = value;
        }
        else
        {
            SOP("Stack Overflow!");
        }
    }
    public Student pop()
    {
        Student value = null;
        if (top > -1)
        {
            value = arr[top];
            top--;
        }
        else
        {
            SOP("Stack Underflow");
        }
    } return value;
```

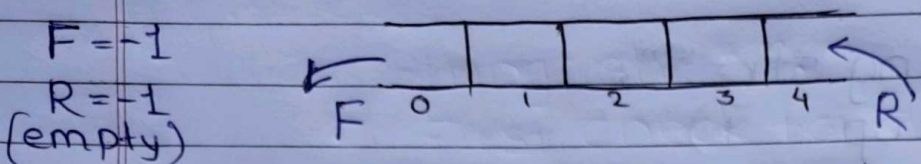
```
public String toString()
{
    String str = " ";
    if (top == -1)
    {
        str = " Stack is Empty! ";
    }
    else
    {
        for (int j = top; j >= 0; j--)
        {
            str = str + arr[j];
        }
        str = str + " \n ";
    }
    return str;
}
}
//Main
public class Main
{
    public static void main(String[] args)
    {
        Stack s1 = new Stack();
        Student p = new Student (" Ram ", 14);
        Student q = new Student (" Sham ", 23);
        s1.push(p);
        s1.push(q);
        s1.push(p);
        s1.push(p);
        SOP(s1);
    }
}
```

Day-2

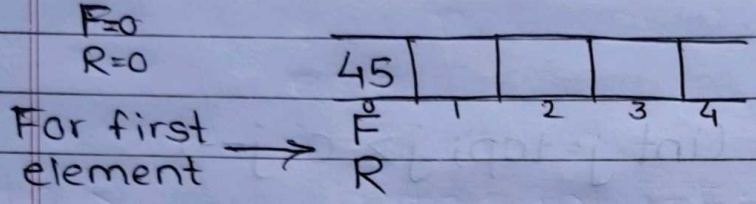
PAGE No.

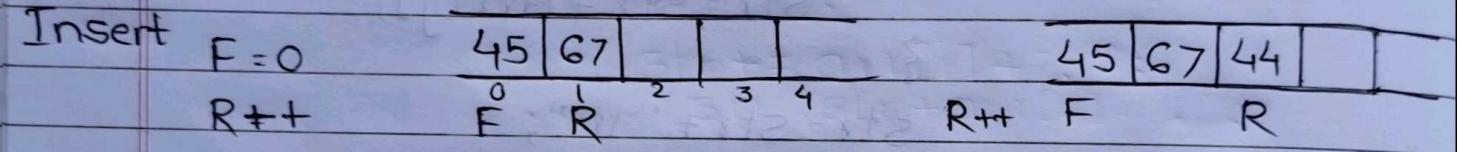
DATE 2/10/21

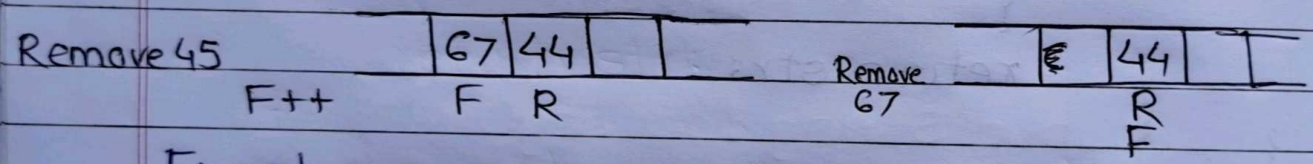
Queue - FIFO (First In First Out)



Front points to the ~~first~~ -1 at initial state
- Front points to the element who will pop firs
- Rear points to the last element in queue







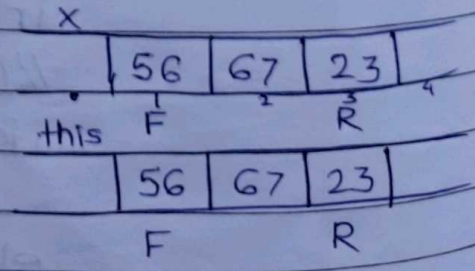
Front never cross Rear

```
// Queue
public class Queue
{
    int [] arr;
    int F;
    int R;
    Queue() // Constructor
    {
        arr = new int [5];
        F = -1; // F = R = -1;
        R = -1;
    }
```

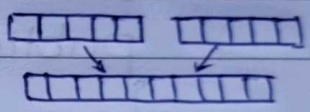
* Queue(int size) //Constructor

```
{
    arr= new int [size];
    F= -1;
    R= -1;
}
public void insert (int value)
{
    if (R<arr.length-1)
    {
        R++;
        arr[R] = value;
        //Check for 1st value insertion
        if (F==-1)
            F=0;
    }
    else
        SOP("Queue is FULL!");
}
public int remove()
{
    //First check for empty condition
    if(F==-1)
    {
        SOP("Queue is Empty!");
        return 0;
    }
    int value = arr[F];
    //Check if this is was the last value.;
    if(F==R)
        F=R=-1;
    else
        F++;
    return value;
}
```

```
public String toString()
{
    String str= " ";
    //First check for empty condition
    if(F==-1)
    {
        str="Queue is Empty!";
    }
    else
    {
F to R      for(int i=F; i<=R ; i++)
            {
                str+= arr[i] +", ";
            }   //str= str+arr[i]+", ";
    }
    return str;
}
public void sameAs(Queue x)
{
    //q2 → this
    //q1 → x
    this.F = x.F;
    this.R = x.R;          //array starts from 0
F to R  for(int i= x.F;(i!= -1 && i<= x.R); i++)
    {       F starts from -1 (when array is empty)
        this.arr[i] = x.arr[i];
    }
}
}
//Main
public class Main
{
```



```
public static void main(String[] args)
{
    Queue q1 = new Queue();
    Queue q2 = new Queue();
    q2.insert(33);
    q2.remove();
    q2.insert(24);
    q2.insert(36);
    q2.insert(48);
    q2.remove();
    q2.sameAs(q1);
    SOP(q2); // Queue is Empty!
}
}
// Queue q3 = q2.concat(q1);
```



PAGE No
DATE / /

Employee Queue

```
// Employee
public class Employee
{
    private String name;
    private int id;
    private int sal;
    public Employee(int id)
    {
        this.id = id;
        this.sal = 0;
        this.name = " ";
    }
    public Employee(int id, String name, int sal)
    {
        this.id = id;
        this.sal = sal;
        this.name = name;
    }
    public String toString() // Call back function
    {
        String str = " ";
        str = "Emp ID:" + id + "\n Emp Name:" +
              name + "\n Emp Sal:" + sal;
        return str;
    }
}
// EmpQueue - using to Employee class
public class EmpQueue
{
    Employee [] arr;
    int F;
    int R;
```

```
public EmpQueue(int Size) //Constructor   (array)
{
    arr = new Employee[size];
    F = -1;
    R = -1;
}
public void insert(Employee value)
{
    //Check full condition
    if (R == arr.length-1)
    {
        SOP("Queue is Full");
    }
    else
    {
        //we have some space in queue
        //But check if this is first insert
        if(R == -1)
        {
            R++;
            F++;
        }
        else
        {
            R++;
        }
        arr[R] = value;
    }
}
public Employee remove()
{
```

In insert all condition should be with **R**

R is more active in insert

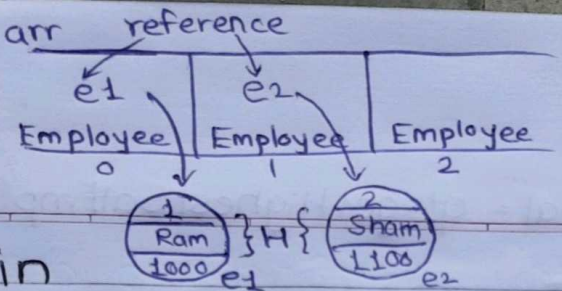
```
if (R == -1)
    F++;
R++;
```
(Fundamentally we can write it like this way)

```
//Check for Empty queue
 if(F==-1)
 {
     SOP("Queue is Empty!");
     return null;
 }
 else
 {
     Employee e=arr[F];
     //check if only one emp in queue?
     if(F==R)
         F=R=-1;
     else
         F++;
     return e;
 }
}
public String toString() //call back function
{
    String str="";
    //First check for empty condition
    if(F==-1)
    {
        str="Queue is Empty!";
    }
    else
    {
        for(int i=F; i<=R; i++)
        {
            str+=arr[i]+"\n";
        }
    }
    return str;
}
}
```

In remove with F

F is more active in remove()

at this location java internally calls toString() method of Employee class



← Holding 3 employees

PAGE No.
DATE / /

Main using EmpQueue

```
//Main
import java.util.Scanner;
public class Main
{
    public static void main(String [] args)
    {
Queue obj       EmpQueue q1 = new EmpQueue(3); //Obj
                            size      creation
Employee obj    Employee e1 = getNewEmpWithUserInput();
                q1.insert(e1);
                Employee e2 = getNewEmpWithUserInput();
                q1.insert(e2);  Employee e3 = new Employee(1,
                                "Pooja", 11000);
                SOP(q1);
                q1.remove();
                S.O.P();
                SOP(q1);
    }
    public static Employee getNewEmpWithUserInput()
    {
        Scanner sc=new Scanner(System.in);
        int sal,id;
        String name;
        SOP("Enter ID:");
        id=sc.nextInt();
        SOP("Enter Name:");
        name=sc.next();
        SOP("Enter Sal:");
        sal=sc.nextInt();
        Employee e=new Employee(id,name,sal);
        return e;
    }
}
```

Day-3

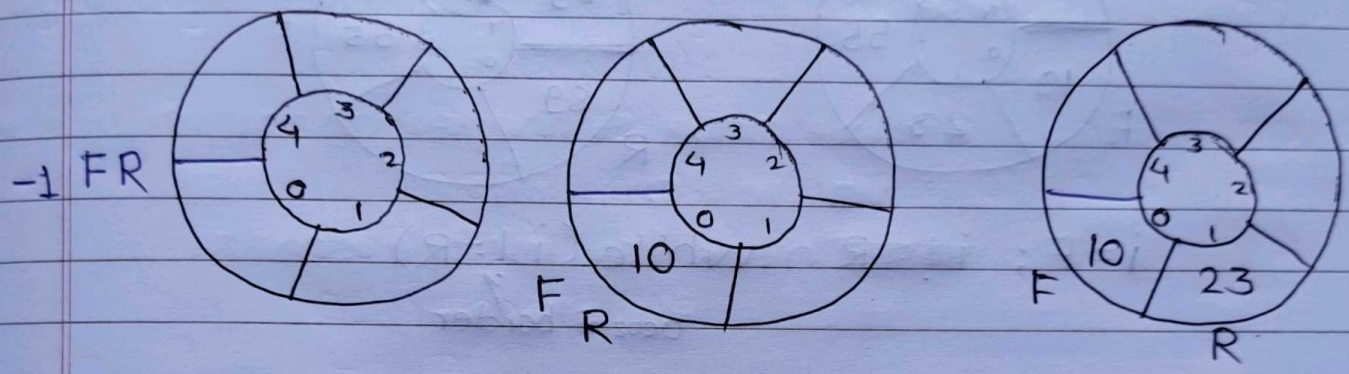
PAGE No.

DATE 3/10/21

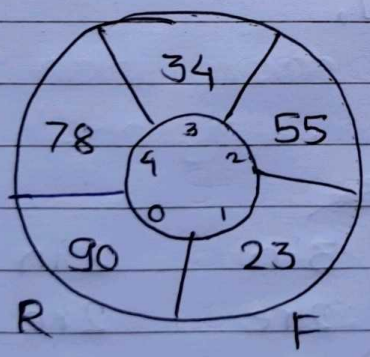
Here only 1 element can be inserted in linear queue

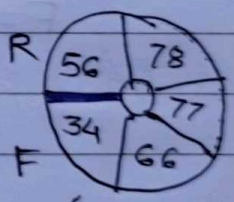
Solution-

* Circular Queue-



If you are on 4th index do not R++ (Out of bound)
then index start from 0.
So start from 0th index after last index





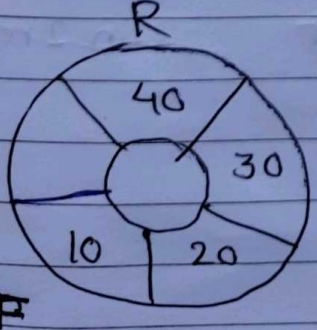
① Full → R+1 == F *

& If R is at 4th & F is at 0th then full condition is

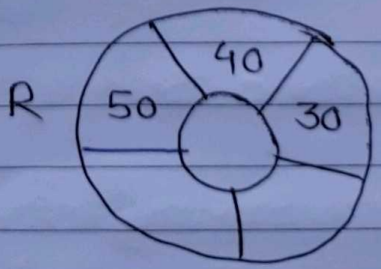
Full → F == 0 && R == len-1 *

②

* Insert-



R++



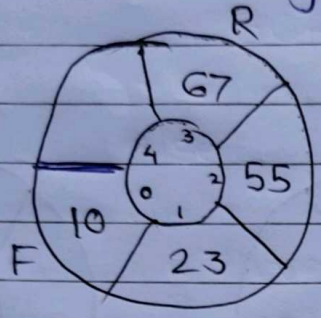
check R == arr.length-1:

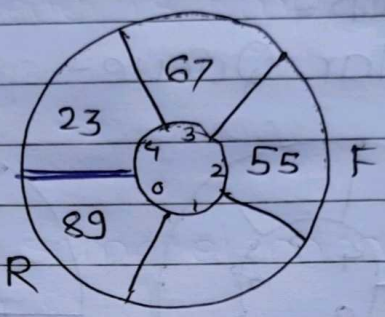
R = 0

PAGE No.
DATE / /

For 1st element - if (F == -1)
F++
now insert value → arr[R] = value

* Print (toString) -



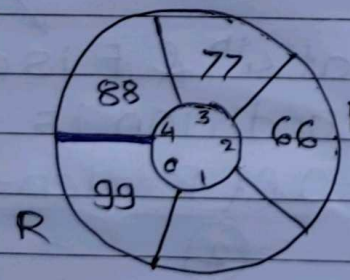


F == -1
Queue empty

i = F; ~~i != R~~ while (i != R)
beoz of border

* Remove -
F == -1 → Queue empty

F == R → value = arr[F];
F = R = -1;



→ 6 ins ←

5 in	3 in	2 in } empty ∴ -1
2 re	2 re	2 re } then 4 insert
1 in	3 in	4 in
Possible to get this case 1 element रहता		not possible 2 in 2 re → -1 हो जाता

```
//CQueue
public class CQueue{
{
    int [] arr;
    int F;
    int R;
    public CQueue()
    {
        arr=new int[5];
        F=R=-1;
    }
    public CQueue(int size)
    {
        arr= new int[size];
        F=R=-1;
    }
    public void insert(int value)
    {
        if((F==0 && R=arr.length-1) || (R+1==F))
        {
            SOP("Queue is Full");
        }
        else
        {
            //we have some space in CQueue
            if(R== arr.length-1)
                R=0;
            else
                R++;
            //now insert element in queue
            arr[R]=value;
            //Check if this is the first element in Queue
            if (F==-1)
                F++;
    }   }
```

```
public int remove()
{
    int value=0;
    if(F==-1)
    {
        SOP("Queue is Empty");
    }
    else
    {
        //we have some element in queue
        value=arr[F]
        //Check for last element
        if(F==R)
        {
            F=R=-1;
        }
        else
        {
            //we have more than 1 element
            if(F==arr.length-1)
                F=0;
            else
                F++;
        }
    }
    return value;
}
public String toString()
{
    String str=" ";
    if(F==-1)
    {
        str="CQueue is empty!";
    }
```

```
for(int j=0;j<50;j++)
{
    SOP(j%5);
}
```

O/p → 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 ....
border

PAGE No. DATE / /

```
        else
        {
            int i=F;
            while(i!=R)
            {
                str=str+arr[i]+", ";
                if(i==arr.length-1)
                    i=0;
                else
                    i++;
                //i=(++i % arr.length);
            }
            //to capture last element
            str=str+arr[i]+", ";
        }
        return str;
    }
}
// Main
public class Main{
    public static void main(String [] args)
    {
        CQueue q1=new CQueue();
        q1.insert(10);
        q1.insert(20);
        q1.insert(30);
        q1.insert(10);
        q1.remove();
        q1.insert(20);
        q1.insert(30);
        q1.remove();
        SOP(q1);
    }
}
```

i=++i %5
i=(++i % arr.length)

DS
arr
LL
Tree

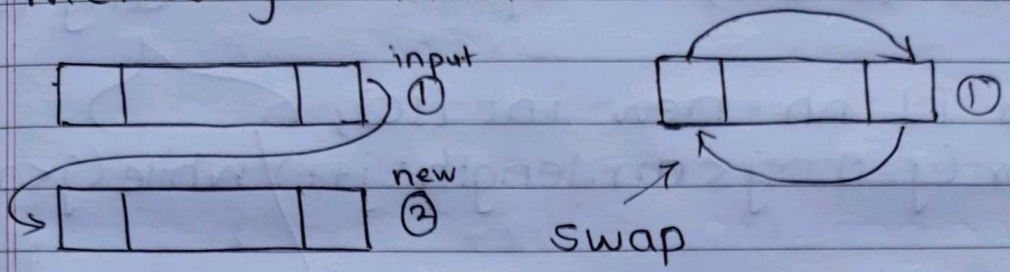
DSA
Stack
queue
Cqueue
map
hashmap

Day-4

PAGE No.

DATE 5/10/21

Memory → RAM



Space - Big O (Depends on heap)

memory allocation (malloc/new)

```
fun(int [] arr)
{
    for(int j=0; j<arr.length;j++)
    {
        int [] a = new int [10];
    }
}
```

Time O(n)

Space O(n) → no of units

```
fun(int [] arr)
{
        // allocation is 1 time
    int [] a = new int [10];
    for(int j=0; j<arr.length; j++)
    {
        . . .
    }
}
```

Time O(n)

Space O(1)

```
fun(int [] arr)
{
    for(int j=0; j<arr.length; j++)//n
    {
        for(int k=0; k<arr.length; k++)//n
        {
            int [] a = new int [10];
        }
    }
}
```

Time $O(n^2)$

Space $O(n^2)$ → nxn

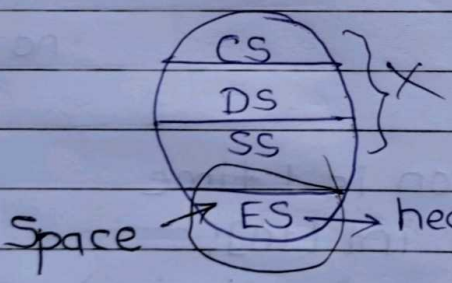
* If type is not mention then default is Time Complexity

PAGE No.
DATE / /

Time — If no loop → O(1)

```
fun(int [] arr)        Space- O(1)        any loop
{
    int [] hh = new int [10];
    for (j=0; j<arr.length; j++)  /  while (j<arr.length)
Time {                                   {
O(2n)
    }                                    }
2 loops  for (k=0; k<arr.length; k++)
∴ 2n     {

    }
}
```



time ∝ number of input (suppose n)

O(n) → 1 loop
O(2n) → 2 loop
O(n²) → 1 loop inside another loop ∴ n×n = n²

* Sorting -

Selection	Quick	Heap
Insertion	Merge	(requires tree)
Bubble		

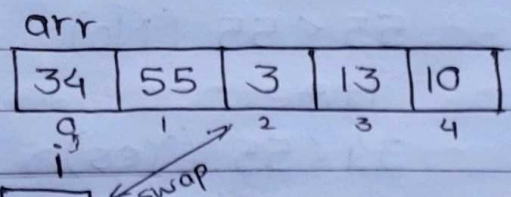
* Selection sort - प्रत्येक element ला, त्याचा proper select करून जागेवर नेवून ठेवायच.

* Insertion - Specific ठिकाणी कुठला element बसतो.

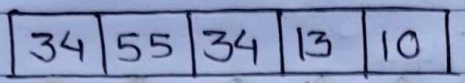
Binary search requires sorted array.

PAGE No. / DATE / /

* 1) Selection Sort - Select 1 ele & put on proper position

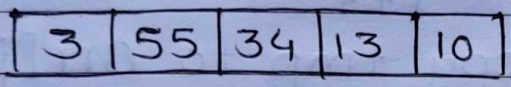


tmp → 34 Increment 'i' till tmp > arr[i] then swap temp & arr[i] increment i compare tmp & arr[i]



tmp 3

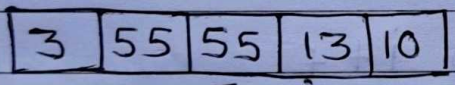
i goes outside loop (put 3 at proper pos)



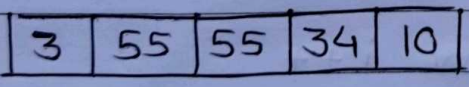
Now,

& i start from 55

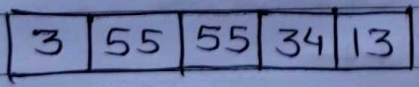
tmp 55 compare ~~arr~~ temp > arr[i] Swap temp & arr[i] i++



tmp 34 swap 34 > 13 ∴ swap i++

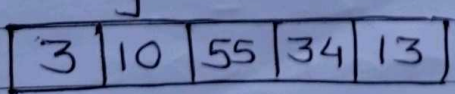


tmp 13 13 > 10 ∴ swap i++

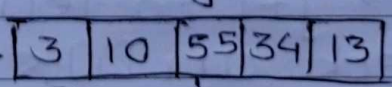


tmp 10

place 10 at location

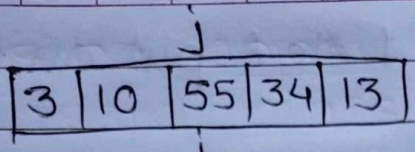


Now j++ →

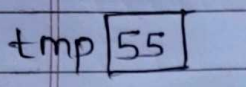


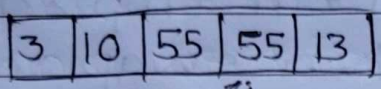
Assign i with j →

j – for place (for each j, i will perform iteration)
i – for iteration

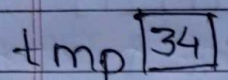


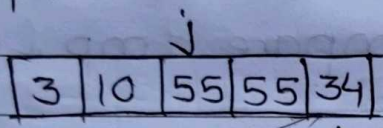
55 < 55 (No)
i++
34 < 55 (Yes)
Swap both



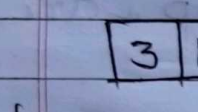


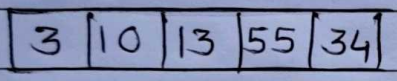
i++
13 < 34
Swap both



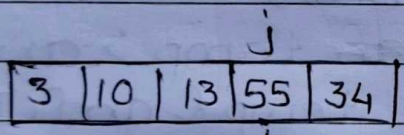


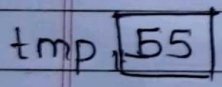
i++
copy tmp to arr[j]



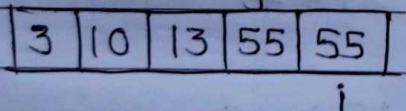


j++
assign j = i
copy 55 to tmp

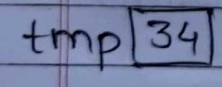


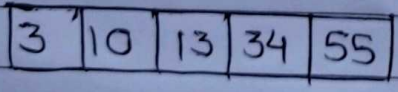


55 < 55 (No) i++
34 < 55 (Yes) Swap both



Now i++
then copy tmp to arr[j]





j++

1. j → 0 to len-1
2. temp → arr[j]
3. i → j+1 to len
4. check arr[i] < temp
5. Swap arr[i] & temp
6. arr[j] = temp

In complexity constanst are negligible
$O(2n) \rightarrow O(n)$
$O(n-1) \rightarrow O(n)$

PAGE No.
DATE / /

```
public class Main{
    public static void main(String[] args)
    {
        int [] arr = new int[] {2,45,-3,67,
                                67,78,42,10,342};
        printArray(arr);
        selectionSort(arr);
        printArray(arr);
    }
    public static void selectionSort(int [] arr)
    {                          j already sorted at the end
        for(int j=0; j<arr.length-1 ; j++)
        {
            int temp = arr[j];
            for(int i=j+1;i<arr.length; i++)
            {
                if(arr[i] < temp)//Asecending
                {
                    int x = temp;
                    temp = arr[i];
                    arr[i] = x;
                }
                arr[j] = temp;
            }
        }
    }
    public static void printArray(int [] arr)
    {
        for(int i=0; i<arr.length; i++)
        {
            SOP("arr[i]+", ");
        }
        SOP();
    }
}
```

(n-1) x n
$O(n^2)$

i=j+1
compare next element
(no need to compare
with itself)
i check decreases

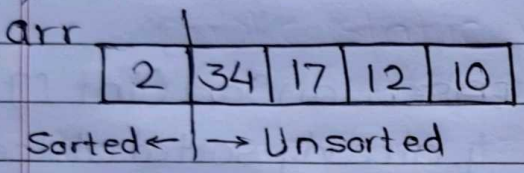
if we write static function in sortclass, no need to create the sort class object in main class.
we can call function used class name itself.
If static is not used then create object in main and call functions.

Using object oriented way it can be

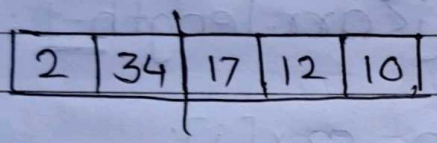
1. Create class Sort
2. Write selection() method in Sort class
3. Call Sort.selection(arr) in main()
4. print() in main

- Can be used for all sortings.

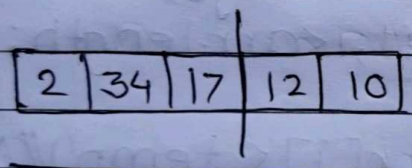
*2) Insertion Sort -



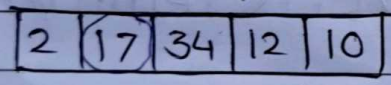
Check 2 fit on which locati (Sorted)



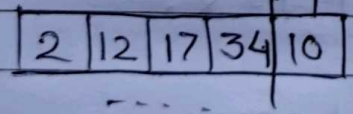
Check 34 fit on which locatio (Sorted)

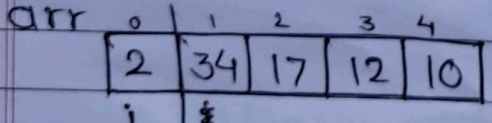


Check 17 fit on which locatio

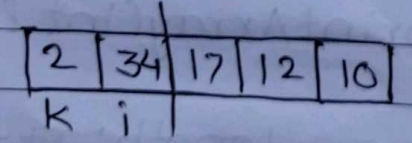


Check 12 fit on which loc

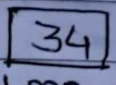


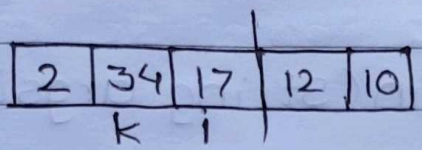


1. Consider arr[0] is sorted
i++



tmp = arr[i];
34
k = i-1;
check k >= 0
check 34 < 2 (No)
arr[k+1] = temp
break;
k-- (k=-1)





i++

tmp = arr[i]

k = i - 1 (2)

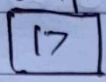
check k >= 0

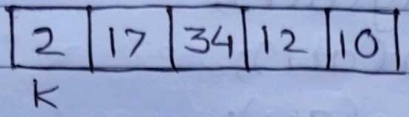
check 17 < 34 (arr[k])

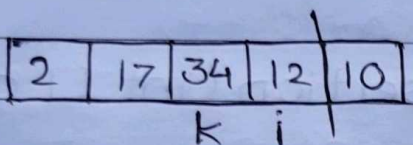
arr[k+1] = arr[k]
(34)

k--

tmp







i++

tmp = arr[i]

k = i - 1

check k >= 0

check 12 < 34

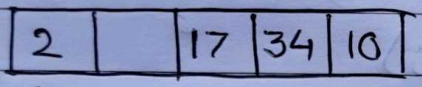
arr[k+1] = arr[k]
(34)

k--

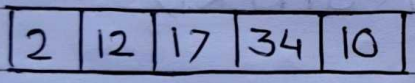
check 12 < 17

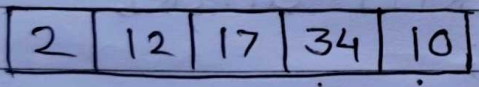
arr[k+1] = arr[k]

k--



arr[k+1] = tmp
(12)



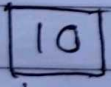


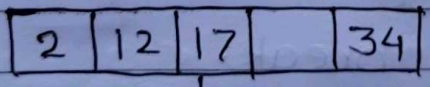
k >= 0

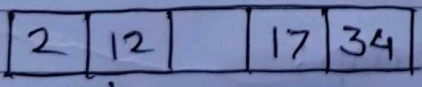
check 10 < 34

arr[k+1] = arr[k]
(34)

k--



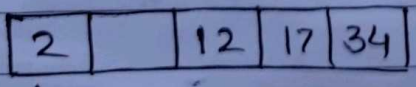




check 10 < 17

arr[k+1] = arr[k]
(17)

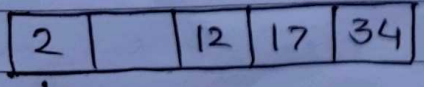
k--



check 10 < 12

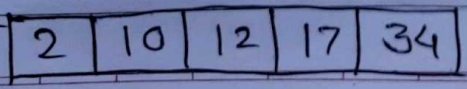
arr[k+1](12) = arr[k]

k--



check 10 < 2 → break;

~~while(k>=0) → false~~



arr[k+1] = tmp

PAGE No.
DATE / /

```
public class Main{
    public static void main(String args[])
    {
        int [] arr= new int [] {2,45,-3,67,67,78,42,10,
                                34,27};
        printArray(arr);
        insertion(arr);
        printArray(arr);
    }
    public static void insertion(int [] arr)
    {
n-1     for(int i=1; i<arr.length; i++)
        {
            int tmp= arr[i];
            int k = i-1;
 n          while(k>=0)
            {
O(n^2)          if (tmp<arr[k])
                {
                    arr[k+1] = arr[k];
                    k--;
                }
                else
                    break;
            }
            arr[k+1] = tmp;
        }
    }
```