Unit-5 Sorting and Searching.

*Searching Algorithom:-

1. Linear Search:-

```
int linear search (int a[], int n, int flag)
{
    int flag = 0;
    for (i = 0; i<n ; i++)
    {
        if (a[i] == key)
        {
            flag = 1;
            Printf ("%d found Pos %d", key, i+1
            return flag;
        }
    }
    return flag;
}
```

→ The best case effenciy is O[1].
→ the wearst case effenciy is O[o].

2. Binary Search:-

key = 11

11, 23, 25, 30, 87, 44, 100

l = 0

u = 6

mid = (0+6)/2 = 3

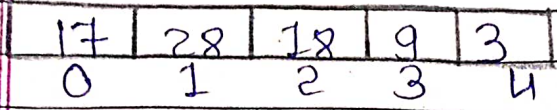
Logic:-

```
int BinearySearch(int a[], int n, int key)
{
  int flag=0, mid, l, u;
  l=0;
  u=n-1;
  mid=(l+u)/2;
  while(l<=u && a[mid]!=key)
  {
    if(a[mid]>key)
    {
      u=mid-1;
    }
    else
    {
      l=mid+1;
    }
    mid=(l+u)/2;
  }
  if(a[mid]==key)
  {
    flage=1;
  }
  else
  {
    flage=0;
  }
  return flag;
}
```

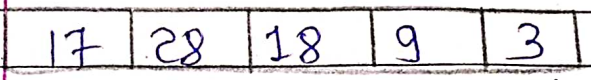
Date: / /
Page No.

* Sorting :-

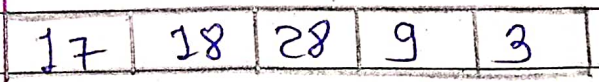
1. Bubble Sorting :-



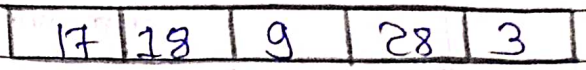
17 < 28 (no)



28 > 18 (swap)

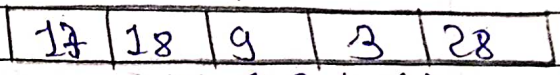


28 > 9 (swap)

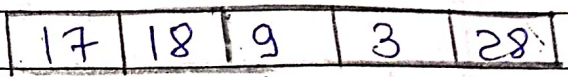


28 > 3 (swap)

Pass-1



17 < 18 (no)



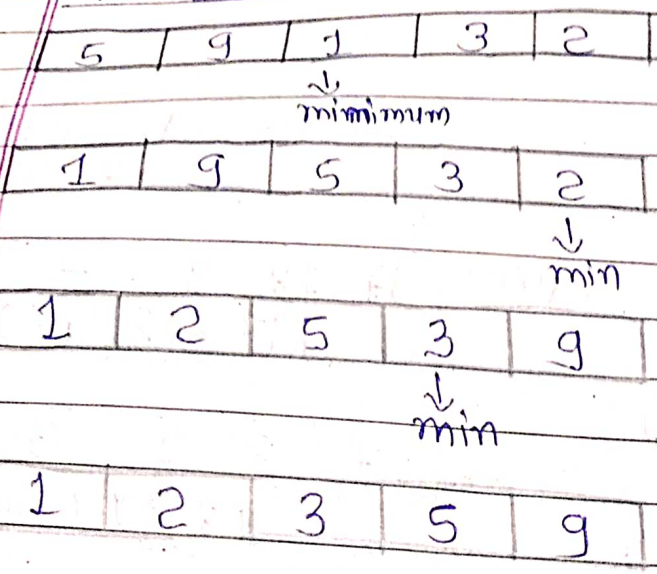
18 > 9 (swap)

→ Same as above.

Logic :-

```
for (i=0; i<n-1-i; i++)
{
    if( a[i] > a[i+1] )
    {
        tmp = a[i];
        a[i] = a[i+1];
        a[i+1] = tmp;
    }
}
```

* selection Sort :-



Logic :-

```
for(i=0; i<n; i++)
{
    min_Pos = i;
    for(j=i+1; j<n; j++)
    {
        if(a[j] < a[min_Pos])
        {
            min_Pos = j;
        }
    }
    tmp = a[i];
    a[i] = a[min_Pos];
    a[min_Pos] = tmp;
}
```

Date: / /
Page No.

3. Merge Sort:

A	10	11	19	30	57
	i	i	i	i	i

B	5	6	15	20	25	67
	i	i	i	i	i	j

C	5	6	10	11	15	19	20	25	30	57	67
	k	k	k	k	k	k	k	k	k		

Logic:-

```
i = 0;
j = 0;
k = 0;
while (i<n && j<m)
{
    if ( a[i] < b[j] )
    {
        c[k] = a[i];
        i++;
        k++;
    }
    else
    {
        c[k] = b[j];
        j++;
        k++;
    }
    while (i< n)
    {
        c[k] = a[i]; i++; k++; }
```

```
while (j<m)
{
  c[k] = b[j];
  j++;
  k++;
}
}
```

4. Insertion Sort:-

70	80	30	10	20
i	j			

tmp = a[j];
if(tmp< a[i])
80 < 70 // false.

70	80	30	10	20
	i	j		

if (tmp < a[i])
30 < 80
a[i+1] = a[i];
tmp = a[j];

70	30	80	10	20
i	j			

j--;
tmp = 30
tmp < a[i] a[i+1] = a[i];
30 < 70

30	70	80	10	20
		i	j	

tmp = 10 a[i+1] = a[i];
tmp < a[i] = 10 < 80

30	70	10	80	20
	i	j	i	

tmp < a[i]
10 < 70
a[i+1] = a[i]; i--;

30	10	70	80	20
i	j			

tmp < a[i]
10 < 30
a[i+1] = a[i]; i--;

10	30	70	80	20
			i	j

tmp < a[i]
20 < 80
a[i+1] = a[i]; i--;

10	30	70	20	80
		i	j	

tmp < a[i]
20 < 70 a[i+1] = a[i]; i--;

10	30	20	70	80
	i	j		

tmp < a[i]
20 < 30
a[i+1] = a[i]; i--;

10	20	30	70	80

Page No.

Logic:

```
for(i = 1; i<n; i++)
{
    i = j - 1;
    tmp = a[j];
    while(i >=0 && tmp < a[i])
    {
        a[i+1] = a[i];
        i--;
    }
    a[i+1] = tmp;
}
```

*Radix Sort / Bucket Sort:

Pass-1:

Find max element:

```
for(i = 1; i<n; i++)
{
    if(a[i] > max)
    {
        max = a[i];
    }
}
Printf(max);
```

Find no of digits:

```
while(no>0)
{
    no = no / 10;
    c++;
}
Printf(c);
```

Date: / /
Page No.

0	1	2	3	4	5	6
1235	111	80	19682	10	560	235

	0	1	2	3	4	5	6	7	8	9
0						1235				
1		111								
2	80									
3			19682							
4	10									
5	560									
6						235				
7										

Pass-1 :-

0	1	2	3	4	5	6
080	010	560	111	19682	1235	235

	0	1	2	3	4	5	6	7	8	9
0	080								080	
1	010	010								
2	560						560			
3		111								
4									19682	
5				1235						
6				235						

Pass-2 :-

0	1	2	3	4	5	6
010	111	1235	235	560	080	19682

	0	1	2	3	4	5	6	7	8	9
0	010									
1		111								
2			1235							
3			235							
4										
5	080					560				
6							19682			

Pass-3.

0	1	2	3	4	5	6
10	80	111	1235	235	560	19682

	0	1	2	3	4	5	6	7	8	9
0	10									
1	80									
2	111	111								
3		1235								
4	235									
5	560									
6		19682								

Pass-4

0	1	2	3	4	5	6.
10	80	111	235	560	1235	19682

→ After find max and number of digits.

```
for(j=1; j<=C; j++)
{
   for(i=0; i<10; i++)
   {
      for(k=0; k<n; k++)
      {
         Bucket[i][j]=0;
      }
   }
}
```

```
for( i = 0; i < n; i++)
{
    x = a[i]/(int) pow (10, j);
    if(j != 1)
    {
        x = x % pow(10, j-1);
    }
    bucket[x][i] = a[i];
}
bucket[x][i] = a[i];
printf("\n");
for(i = 0; i < 10; i++)
{
    if( bucket[i][j] != 0)
    {
        a[k] = bucket[i][j];
        k++;
    }
}
for(i = 0; i < n; i++)
{
    printf("%d", a[i]);
}
printf("\n");
}
}
```

Date: / /
Page No.

* Quick Sort :-

Left → Big | Small → Right

5	37	26	15	12	92	86	33
p	i						j
j	j						

5	37	26	15	12	92	86	33
	p				i		j

5	37	26	15	12	33	86	92
	p				j	i	
						j	

5	33	26	15	12	37	86	92
	p			j		p	j

5	12	26	15	33	37	86	92
	p	i					
	j	j					

5	12	26	15	33	37	86	92

5	12	26	15	33	37	86	92
	p	i					
	j						

5	12	26	15	33	37	86	92
		p					
		j					

5	12	15	26	33	37	86	92

Date: / /
Page No.

Logic:-

```
{
int pivot, i, j, tmp;
if( first < last)
{
  pivot = first;
  i = first;
  j = last;
while (i<j)
{
  while (a[i] <= a[pivot] && i < last)
  i++;
  while (a[j] > a[pivot] && j > first)
  j--;
  if(i<j)
  {
    tmp = a[i];
    a[i] = a[j];
    a[j] = tmp;
  }
}
tmp = a[pivot];
a[pivot] = a[j];
a[j] = tmp;

quicksort(a, first, j-1);
quicksort(a, j+1, last);
}
```

* Stack :-

LIFO (Last in first out)

-> In stack array elements are insert and last element is first out so It is a last in first out methods.

* function :-

1. PushH ( ) -> Insert.
2. PoP ( ) -> Delete.
3. PeeP / Peek ( ) -> Show current PoP value.
4. display ( ) -> display.

-> In stack top element are always +1 start with -1 ToP / Tos.

-> Stack is a collection of data item that can be accessed at only one end caved top.

-> Insert and delete in a stack only at the toP.

```
insert = -1   top ++ ;
Delete = toP- - ;
```

ToP = -1 that means empty stack.

E.x. Bangle in a hands.

Date: / /
Page No.

*Logic:

```
#define size 5
int stack[size];
int top = -1
void main()
{
    int choice;
    clrscr();
    Printf("\n\n\t 1. Push");
    Printf("\n 2. Pop");
    Printf("\n 3. Peep");
    Printf("\n 4. Display");
    Printf("\n 0. Exit");
    Printf("Enter choice");
    Scanf("%d", &choice);

    Switch (choice)
    {
        Case 1;
            // Push
            break;
        Case 2;
            // Pop
            break;
        Case 3;
            // Peep
            break;
        Case 4;
            // display
            break;
        Case 0;
            exit(0);
```

```
        default :
                Printf ("\n Enter Choice");
    } while (choice != 0);

Case 1:
    void Push ()
    {
      if (top == size-1)
      {
        Printf ("stack overflow / full");
      }
      else
      {
        Printf ("\n Enter value ...");
        Scanf ("%d", &val);
        top ++;
        stack [top] = val;
      }
    }

Case 2:
    void display ()
    {
      if (top == -1)
      {
        Printf ("Stack empty");
      }
      else
      {
        for (i = top; i >= 0; i--)
        {
          Printf ("\n %d", stack [i]);
        }
    } } }
```

Date: / /
Page No.

Case 3:

```
Void Pop()
{
    if (top == -1)
    {
        Printf ("empty");
    }
    else
    {
        Printf ("\n %d delete", stack[top]);
        top--;
    }
}
```

Case 4:

```
void Peep()
{
    int Pos;
    if (top == -1)
    {
        Printf ("\n stack empty");
    }
    else
    {
        Printf ("Enter Position you are went to
                see ... ");
        Scanf ("%d", &Pos);
        if (Pos > top + 1)
        {
            Printf ("Position is higher there element");
        }
```

```
else if (top - pos + 1 < 0)
{
   Printf ("\n\n pos is lower than element");
}
else
{
   Printf("\n\n %d is at position %d",
      stack[top - pos + 1], pos);
}
}
```

* Application of Stack:-

Evaluation of Arithmatic expression / Polish Notation.

1. Infix.
2. Postfix.
3. Prefix.

Infix = a + b (1 operator in between two operands).

Convert expression
a + b * c / d - f

Postfix :- a b + (1 operator are behind two operands.)

Prefix :- + a b (1 operator front of two operand).

In → between
Post → After.
Pre → before

* Priority of execution:-

1. [ ], { }, ( ).
2. ^, ↑, √
3. /, *
4. +, -

e.x. a+b*c/d-f convert in Postfix.

a+b*c/d-f

a+b* cd/ -f

a+ bcd/* -f

abcd/*+ -f

Ans:- abcd/*+f-

e.x. {(A+B)*{(C/D)-(E^(f*√))}}

[(A+B)*((C/D)-(E^ f√*))]

((A+B)*((C/D)- EF√*^))

((A+B)*(CD/ - EF√*^))

((A+B)* CD/ EF√*^ -)

AB+ CD/ EF√*^ - *

Date: / /
Page No.

e.x. ((A+((B^C)-D))*(E-(A/C)))

((A+(BC^-D))*(E-(A/C)))
(A+(BC^-D))*(E-AC/))
(A+BC^D-)*(E-AC/))
ABC^D-+ * EAC/-)
ABC^D-+EAC/-*

e.x. ((A+B)^C-(D*E)/F)

((A+B)^ C-(D*E) F/)
(AB+^ C- DE* F/)
Postfix. AB+C^ DE*-F/

e.x. ((A+B)^C-(D*E)/F) Prefix.

=(+AB ^C- *DE /F)
=^+ABC -/*DEF
Ans -^* ABC /* DEF.

* Convertt in stack evaluation:-

e.x. A+(B*C)/D

A+ BC* /D
A+ BC* D/

Ans:- ABC* D/+

A=2 B=3 C=4 D=6.

2 3 4 * 6 / +

Date: / /
Page No.

Read Symbol	Stack	Output
2		2
3		2, 3
4		2, 3, 4
*	*	2, 12
6		2, 12, 6
/	/	2, 2
+	+	4.

[Ans = 4]

e.x. 9 - ((3 * 4) + 8) / 4.

Post :
9 - (34 * +8) / 4
9 - (34 * 8+) / 4
9 - 34 * 8+ / 4
[9 34 * 8+ 4 / -]

Symbol	expression	Output
9		9
3		9, 3
4		9, 3, 4
*	*	9, 12
8		9, 12, 8
+	+	9, 20
4		9, 20, 4
/	/	9, 5
-	-	4

[Ans = 4]

Date: / /
Page No.

ex. ((A+B) ^ C - (D * E) / F)

= (AB+C ^ - DE* F /)

(= AB+C ^ DE * F / -)

Symbol	Stack	Output
(	(	
(	((	
A	((	A
+	((+	A
B	((+	AB
)	(	AB+
^	(^	AB+
C	(^	AB+C
-	(-	AB+C^
(	(-(	AB+C^
D	(-(	AB+C^D
*	(-(*	AB+C^D
E	(-(*	AB+C^DE
)	(-	AB+C^DE*
F	(-	AB+C^DE*F
/	(-/	AB+C^DE*F/-

(Ans: AB+C^DE*F/-)

Date: / /
Page No.

Ex. ((A+B)* ((C/D)-(E^(F*G))))

Symbol	Stack	Output
(	(	
(	((	
A	((	A
+	((+	A
B	((+	AB
)	(	AB+
*	(*	AB+
(	(*(	AB+
(	(*((	AB+
C	(*((	AB+C
/	(*((/	AB+C
D	(*((/	AB+CD
)	(*(	AB+CD/
-	(*(-	AB+CD/
(	(*(-(	AB+CD/
E	(*(-(	AB+CD/E
^	(*(-(^	AB+CD/E
(	(*(-(^(	AB+CD/E
F	(*(-(^(	AB+CD/EF
*	(*(-(^(*	AB+CD/EF
G	(*(-(^(*	AB+CD/EFG
)	(*(-(^	AB+CD/EFG*
)	(*(-	AB+CD/EFG*^
)	(*	AB+CD/EFG*^-
)		AB+CD/EFG*^-*

[Ans:- AB+CD/EFG*^-*]

e.x. A+B - C*D

Post fix :-

A+B - CD*

AB* - CD*

[Ans:- AB+ CD*-]

Pre fix :-

+AB - *CD

[Ans:- -+AB *CD]

Post fix :- AB+CD*- A=2, B=3, C=5, D=1.

23+51*-

Symbol	Stack	Output
2		2
3		2,3
+	+	5
5		5,5
1		5,5,1
*		5,5
-		0

Date: / /
Page No.

-+23*51

Symbol	Stack	Output
1		1
5		1,5
*	*	5
3		5,3
2		5,3,2
+	+	5,5
-	-	0

Logic:-

```
# include <stdio.h>
# include <conio.h>
# define Stack [size];
 int  top = -1;
 void  Push (int val)
{
      top++;
      Stack [top] = val;
 }
 int  top ()
{
      top--;
      return stack [top+1];
}

Void  main ()
{
    char Post [50], tmp [50];
    int val, val1, val2, i;
    clrscr();
```

```
Printf("Enter Postfix expression:-");
Scanf("%s", Post);
Strcpy(tmp, Post);
Printf("\n\t tmp:- %s", tmp);

for(i=0; Post[i]; i++)
{
    if((Post[i]>='a' && Post[i]<='z') ||
    (Post[i]>='A' && (Post[i]<='Z'))
    {
        Printf("\n Enter value for %c:",
                Post[i]);
        fflush(stdin);
        scanf("%c", &tmp[i]);
    }
}
Printf("\n\n temp String %s \n", temp);

for(i=0; Post[i]; i++)
{
    if((Post[i]>='a' && (Post[i]<='z')
    || (Post[i]>='A' && (Post[i]<='Z')
    {
        Push(tmp[i] - 48);
    }
    else
    {
        val2 = PoP();
        val1 = PoP();
        if(Post[i]=='+')
        {
            val = val1 + val2;
        }
```

```
        if (Post[i] == '-')
        {
            val = val1 - val2;
        }
        if(Post[i] == '*')
        {
            val = val1 * val2;
        }
        if(Post[i] == '/')
        {
            val = val1 / val2;
        }
        Push(val);
    }
}
    Printf("Post fix : %s \n", Post);
    Printf("Ans: %d \n", Pop());
    getch();
}
```

e.x. (A - (B / C + (D % E * F) / G) * H.)

Symbol	Stack	Output
A	(	A
-	(-	A
(	(- (	A
B	(- (	A B
/	(- (/	A B
C	(- (/	A B C
+	(- (+	A B C /
(	(- (+ (	A B C /
D	(- (+ (	A B C / D
%	(- (+ (%	A B C / D

E	(-(+(↑	ABC/DE
*	(-(+(*	ABC/DE*↑
F	(-(+(*	ABC/DE↑F
)	(-(+	ABC/DE↑F*
/	(-(+	ABC/DE↑F*/
G	(-(+	ABC/DE↑F*/G
)	(-(	ABC/DE↑F*/G+
*	(-	ABC/DE↑F*/G+*
H	(-	ABC/DE↑F*/G+*H

e.x. (A+B)*C/D

AB+ * C/D
AB+ * CD/
AB+ CD/*

Read symbol	Stack	Output
(	(	
A	(	A
+	(+	A
B	(+	AB
)		AB+
*	*	AB+
C	*	AB+C
/	*/	AB+C
D	*/	AB+CD
D	*/	AB+CD/*

Ans = AB+CD/*

Date: / /
Page No.

* Infix to Postfix:-

```
Char Stack [size];
int top = -1; curPos = -1; lastPos = -1;
void Push (char symbol)
{
    top = top + 1;
    Stack [top] = symbol;
}
Char Pop ()
{
    Char symbol;
    symbol = stack [top];
    top = top - 1;
    Return (symbol);
}
Char read-symbol (char infix [])
{
    curPos = curPos + 1;
    Return (infix [curPos]);
}
int isP (char x)
{
    int value;
    Switch (x)
    {
    case '+';
    case '-';
        value = 2;
        break;
    case '*';
    case '/';
        value = 4; break;
```

```
        case '5';
            value=5;
            break;
        case 'C';
            value=0;
            break;
        default;
            value=0;
            break;
    }
    return (value);
}
int icp(char x)
{
    int value;
    switch.(x)
    {
        case '+';
        case '-';
            value=1;
            break;
        case '*';
        case '/';
            value=3;
            break;
        case '^'
            value=6;
            break;
        case '(';
            value=9;
            break;
        case ')';
            value=0; break;
        default;
            value=7;
            break;
    }
    return (value);
}
```

Date: / /
Page No.

* Queue:

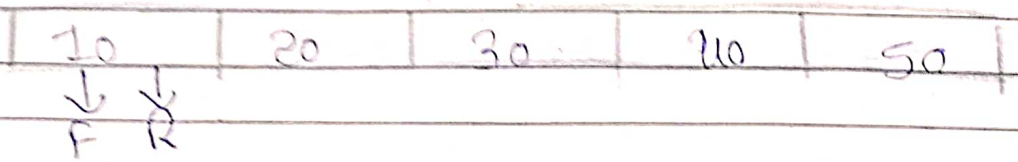
FIFO (First In First Out)
FCFS (First Come First Serve)

Types:-

double simple queue
Circular queue
dequeue queue
Priority queue

If front = -1 and rear = -1 queue is empty.

Simple queue :-



Insert :-

```
Void Insert()
{
  int val;
  if( rear==size-1)
  {
    Printf(" queue is full");
  }
  else
  {
```

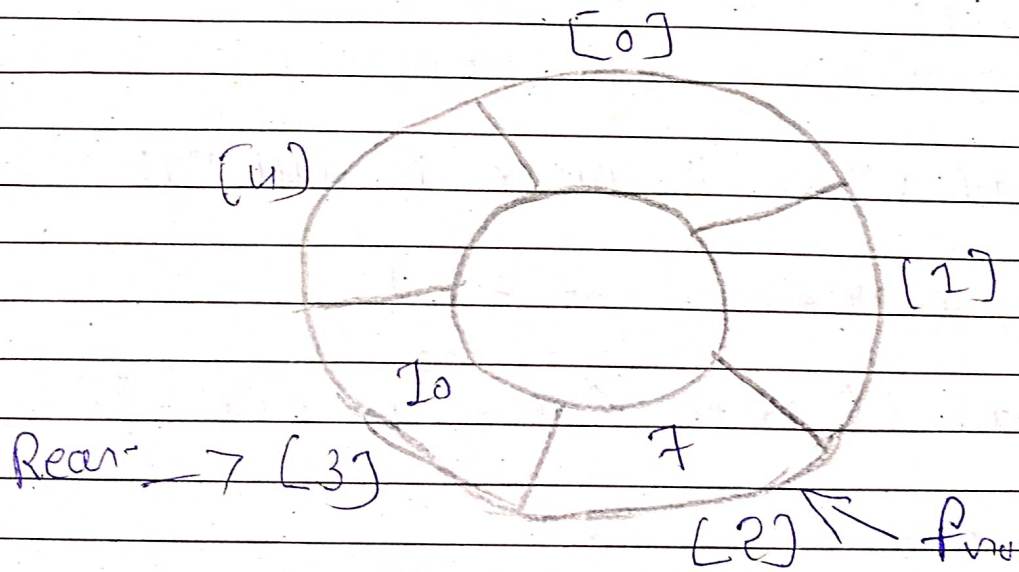
```
        if (front == -1)
        {
            front = 0;
        }
        rear++;
        Printf("\n enter value");
        Scanf("%d", &val);
        queue[rear] = val;
    }
}
```

* Delete :-

```
void delete()
{
    if(front == -1)
    {
        Printf("\n it empty queue");
    }
    else
    {
        if (front == rear)
        {
            front = -1, rear = -1;
        }
        else
        {
            Printf("\n deleted successfully");
            front = front++;
        }
    }
}
```

```
void display()
{
    int i;
    if( front == -1 && rear == -1 )
    {
        Printf ("\n\t empty queue");
    }
    else
    {
        Printf("\n\t Elements of queue");
        for(i=front; i<= rear; i++)
        {
            Printf ("\n\t\t %d \n", queue[i]);
        }
    }
}
```

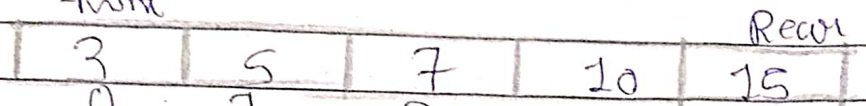
Circular queue :



Rear = front - 1 or front = Rear + 1
It is circular queue.

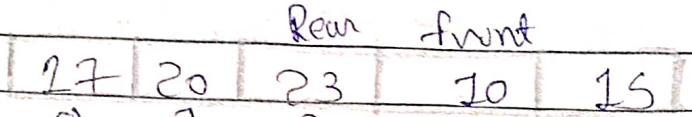
Queue full condition:-

if front = 0 and Rear is at last Position.



or

if front = Rear + 1



* Insert:-

```
void Insert()
{
  int val;
  if (front == 0 && rear == size - 1)
  {
    Printf ("\n; t queue is full");
  }
  else if (rear == front - 1)
  {
    Printf (" int queue is full ");
  }
  else if (front == -1)
  {
    front = 0;
    rear = 0;
    queue [rear] = val;
  }
```

```
else if (rear == size - 1)
{
    rear = 0;
    queue[rear] = val;
}
else
{
    rear++;
    Printf("enter value:");
    Scanf("%d", &val);
    queue[rear] = val;
}
}
```

* Delete:

```
Void deleteq()
{
    if (front == -1)
    {
        Printf("\n\t empty queue");
    }
    else
    {
        if (front == rear)
        {
            front = -1, rear = -1;
        }
        else if (front == rear)
        {
            front = size - 1;
        }
```

```
else
{
    Printf (" deleted successfully ");
    front = front + 1;
}
}
}
```

* display

```
Void display()
{
    int i;
    if (front == -1 && rear == -1)
    {
        Printf (" \n\t empty queue ");
    }
    else if (rear >= front)
    {
        Printf ("\n\n\t Element of queue=>");
        for( i= front; i<= rear; i++)
        {
            Printf (" front : %d \n", front);
            Printf (" rear : %d \n", rear);
            Printf ("\n\t %d \n", queue[i]);
        }
    }
    else
    {
        for (i = front; i<= size-1; i++)
        {
            Printf (" \n\t = %d \n ", queue[i]);
        }
```

Date: / /
Page No.

```
for (i=front ; i<=rear; i++)
{
    Printf("\n%d %d\n", queue[i]);
}
}
}
```