DECEMBER-2022 [END SEM][5925] - 256 | **Solved Paper**

Course 2019

Time : $2\frac{1}{2}$ Hours] **[Max. Marks : 70**

Instructions to the candidates :

1) Answer Q.1 or Q.2, Q.3 or Q.4, Q.5 or Q.6, Q.7 or Q.8.

2) Figures to the right indicate full marks.

3) Neat diagrams must be drawn wherever necessary.

4) Make suitable assumption whenever necessary.

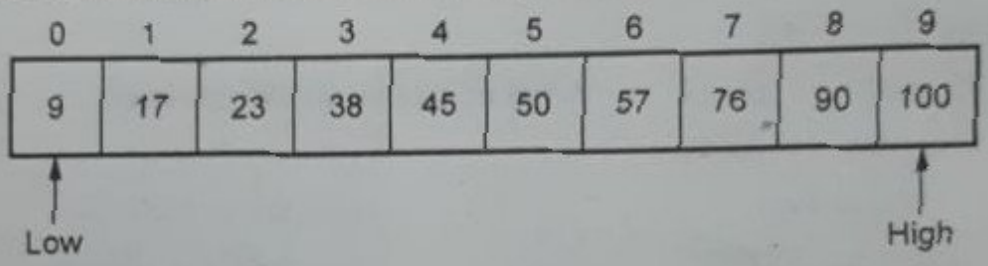
Q.1 a) *Write pseudo 'Python' algorithm (recursive) for binary search. Apply your algorithm on the following numbers stored in array from A[0] to A[10] 9, 17, 23, 38, 45, 50, 57, 76, 90, 100 to search numbers 10 and 100.* **[9]**

Ans. : Recursive Python Program

```
def BinRsearch(arr,KEY,low,high):
    if(high >= low):
        m =(low+high)//2        #mid of the array is obtained
        if(arr[m] == KEY):
            return m
        elif(arr[m]>KEY):
            return BinRsearch(arr,KEY,low,m-1)  #search the left sub list
        else:
            return BinRsearch(arr,KEY,m+1,high) #search the right sub
                                                  list
    else:
        return -1                   #if element is not present in the list

print("\nHow many elements are there in Array?")
n = int(input())
array = []
```

```
i=0
for i in range(n):
    print("\n Enter element in Array")
    item = int(input())
    array.append(item)

print("Resultant array is\n")
print(array)

print("\n Enter the key element to be searched: ")
key = int(input())
location = BinRsearch(array,key,0,len(array)-1)
if(location != -1):
    print("The element is present at index:",location)
else:
    print("\n The element is not present in the list")
```

Let us store the numbers in an array A as -



i) Let, the **KEY = 10**

Now obtain the mid element using following formula -

$$m = (Low + High)/2$$

$$= (0 + 9)/2$$

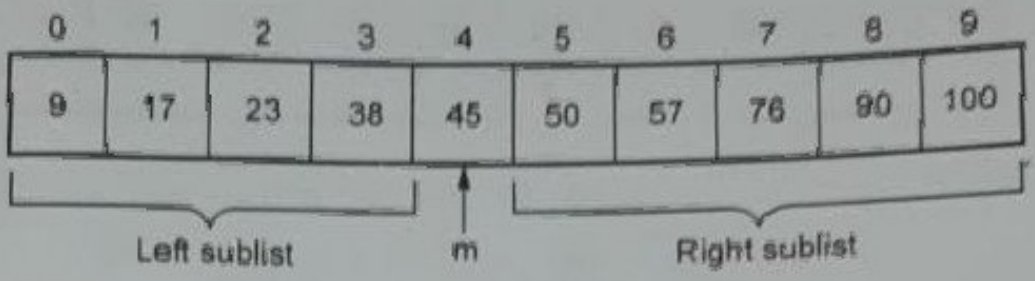
$$m = 4$$

Check if $A[m] \overset{?}{=} KEY$

i.e. if $A[4] \overset{?}{=} 10$

$A[4] = 45$ and $10 < 45$

∴ Search left sublist.

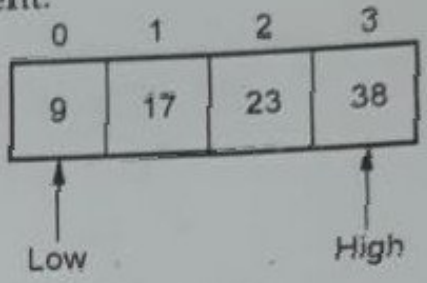


Again divide this list and find mid element.

$m = (Low + High)/2$

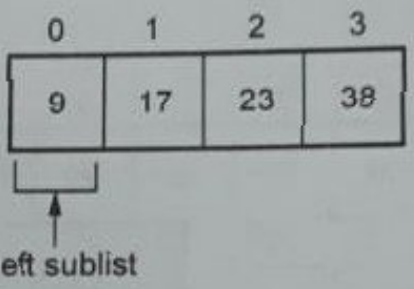
$= (0 + 3)/2$

$m = 1$



A[m] = 17 and 10 < 17

Hence search left sublist.



As there is only single element in the list and it is not equal to 10. That means 10 is **not present** in the list.

ii) Let **KEY = 100**

Now obtain the mid element using following formula -

$m = (Low + High)/2 = (0 + 9)/2$

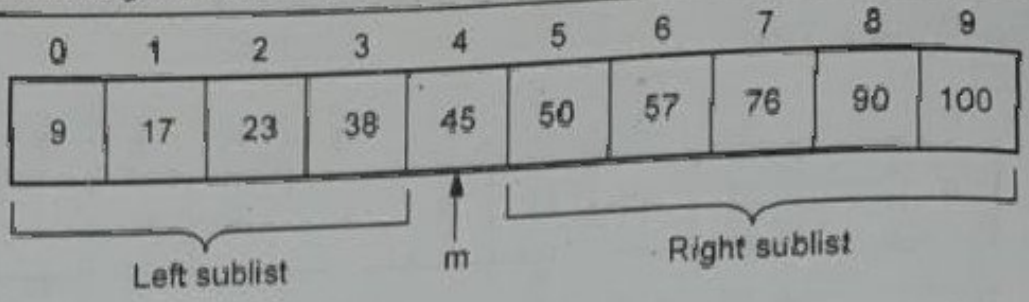
$m = 4$

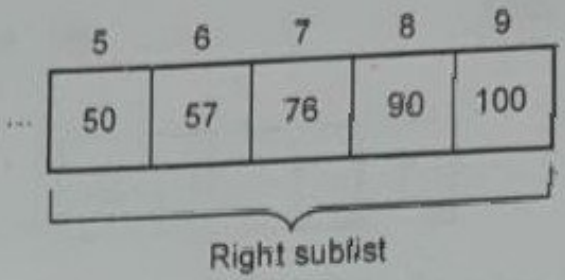
Check if $A[m] \stackrel{?}{=} KEY$

i.e. if $A[4] \stackrel{?}{=} 100$

As A[4] = 45 and 100 > 45.

Hence search right sublist.

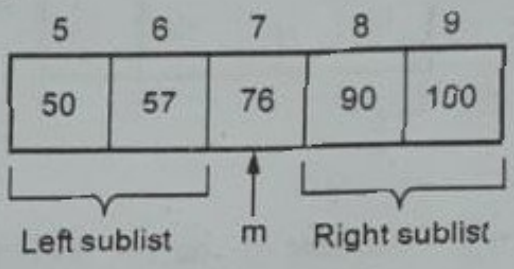




Now obtain mid element

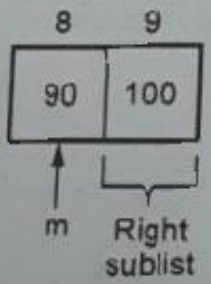
$$m = (Low + High)/2$$
$$= (5 + 9)/2$$
$$= 14/2$$
$$m = 7$$

The A[m] = 76



As A[m] < KEY i.e. 76 < 100

Hence search right sublist.



Here m = 8 and A[m] = 90

90 < 100

∴ Search right sublist

The right sublist contains only one element. i.e. 100

Thus the element 100 is present in array at A[9] location.

b) *Explain the quick sort algorithm. Show the contents of array after every itertion of your algorithm start from following status of array 27, 76, 17, 9, 57, 90, 45, 100, 79.* [9]

Ans. : Refer Q.16 of Chapter - 3.

Consider the first element as pivot element.

27	76	17	9	57	90	45	100	79
↑	↑							↑
Pivot	i							j

If A[i] < pivot then increment i

If A[j] > pivot then decrement j

When we get above conditions false, swap A[i] and A[j]

27	76	17	9	57	90	45	100	79
↑	↑						↑	
Pivot	i						j	

27	76	17	9	57	90	45	100	79
↑	↑		↑					
Pivot	i		j					

Swap A[i] and A[j]

27	9	17	76	57	90	45	100	79
↑	↑		↑					
Pivot	i		j					

27	9	17	76	57	90	45	100	79
↑		↑	↑					
Pivot		j	i					

Swap A[j] and A[Pivot]

After pass 1 :

[17 9] 27 [76 57 90 45 100 79]

These two sublists are sorted recursively.

Consider the list

[17 9]
↑ i/j
Pivot

Swap A[j] and A[Pivot]

After pass 2 :

9 17 27 [76 57 90 45 100 79]
↑ i j
Pivot

If A[i] < pivot then increment i

If A[j] > pivot then decrement j

When we get above conditions false,

Swap A[i] and A[j]

... [76 57 90 45 100 79]
↑ i j
Pivot

Swap A[i] and A[j]

... [76 57 90 45 100 79]
↑ i j
Pivot

Swap A[i] and A[j]

... 76 57 45 90 100 79
Pivot i j
76 57 45 90 100 79
Pivot j i

Swap A[j] and A[Pivot]

... 45 57 76 90 100 79

After pass 3 :

9 17 27 45 [57 76 90 100 79]
↑ i j
Pivot

9 17 27 45 57 [76 90 100 79]

After pass 4 :

... [76 90 100 79]
↑ i j
Pivot

No swapping takes place.

9 17 27 45 57 [76 90 100 79]

After pass 5 :

9 17 27 45 57 76 [90 100 79]
↑ i j
Pivot

Swap A[i] and A[j] from right sublist.

... [90 79 100]
↑ i j
Pivot

... [90 79 100]
↑ j i
Pivot

Swap A[j] and A[Pivot]

... [79 90 100]

After pass 6 :

9 17 27 45 57 76 79 90 100

After pass 7 :

9 17 27 45 57 76 79 90 100

This is a sorted list.

OR

Q.2 a) *Explain in brief the different searching techniques. What is the time complexity of each of them ?* [9]

(Refer Q.3, Q.4 and Q.7 of Chapter - 3)

b) *Write an algorithm of selection sort and sort the following numbers using selection sort and show the contents of an array after every pass - 81, 5, 27, – 6, 61, 93, 4, 8, 104, 15.* [9]

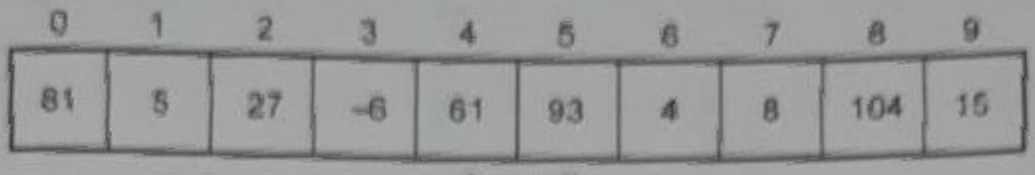
Ans. :

Python Program

```
def SelectionSort(arr,n):
    for i in range(n):
        Min = i
        for j in range(i+1,n):
            if(arr[j]<arr[Min]):
                Min = j
        temp=arr[i]
        arr[i]=arr[Min]
        arr[Min]=temp

    print(arr)

print("\n Program For Selection Sort")
print("\nHow many elements are there in Array?")
n = int(input())
array = []
i=0
for i in range(n):
    print("\n Enter element in Array")
    item = int(input())
    array.append(item)

print("Original array is\n")
print(array)
```

print("\n Sorted Array is")
SelectionSort(array,n)

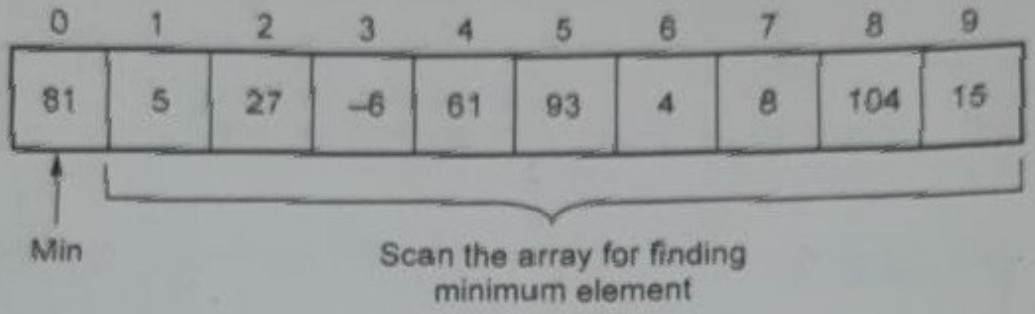
Let



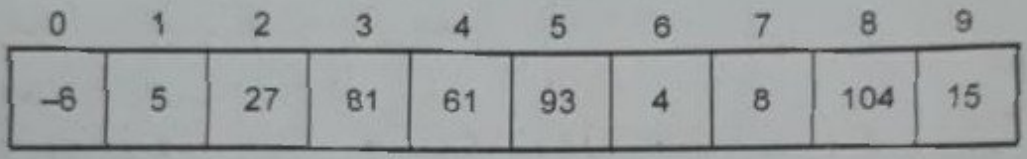
Array A

are the element in array A.

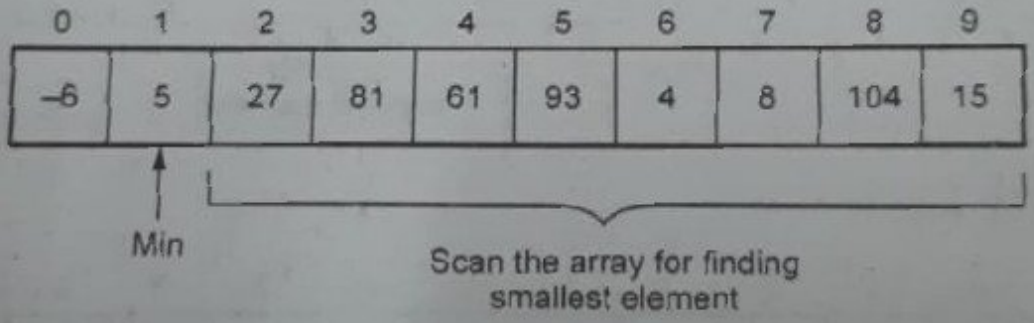
Pass 1 : Consider the A[0] as the minimum element.



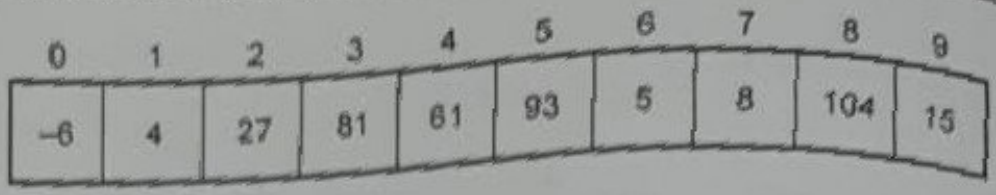
If the smallest element is found then swap it with the A[0] element. Hence swap 81 with – 6.



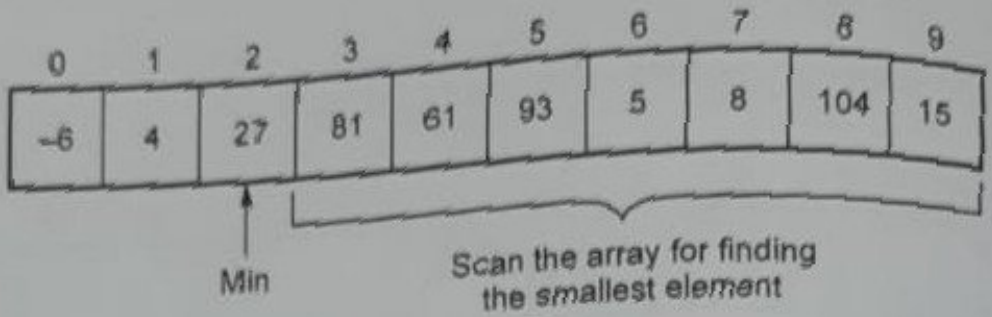
Pass 2 :



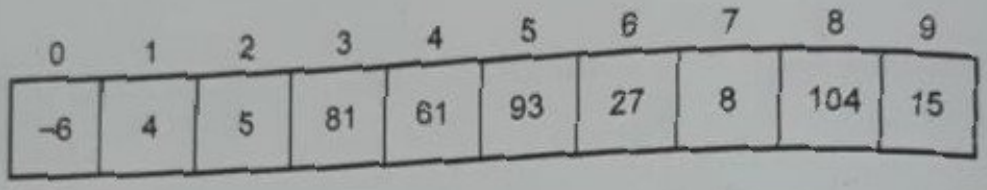
As 4 is the smallest element, swap it with 5.



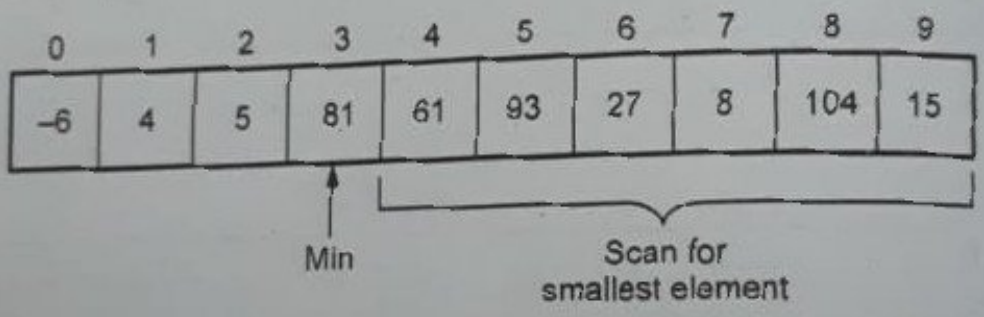
Pass 3 :



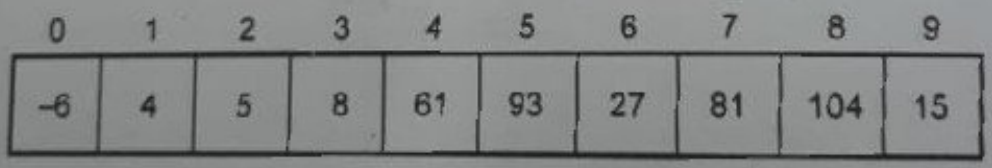
Swap 27 with 5.



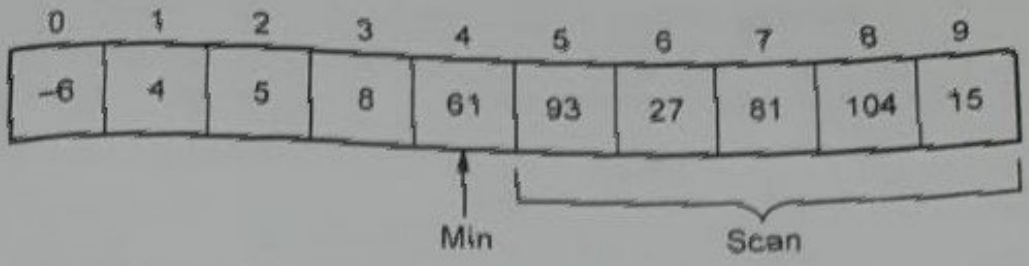
Pass 4 :



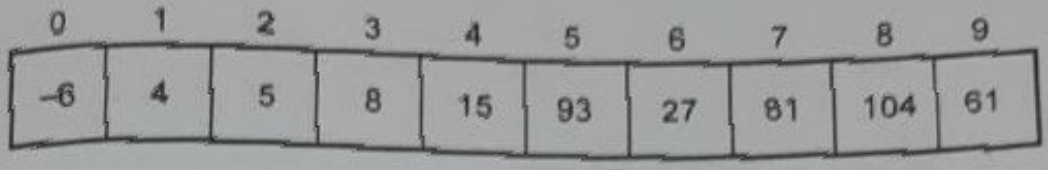
Swap 81 with 8.



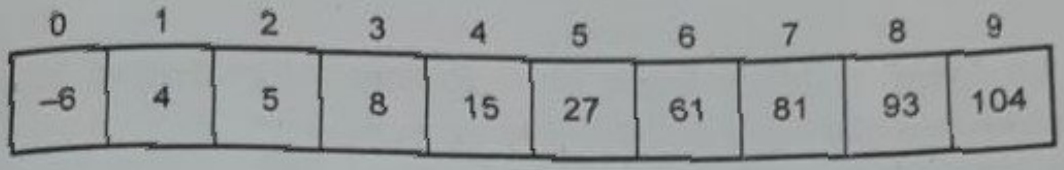
Pass 5 :



Swap 61 with 15. The array will be



Continue in this manner and finally we will get the sorted list as



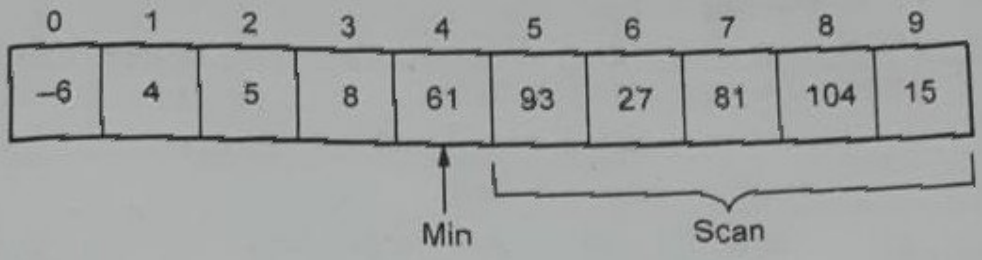
Q.3 a) *What is linked list ? Write a pseudo C++ code to sort the elements.* [9]

Ans. : Linked list : Refer Q.2 of Chapter 2.

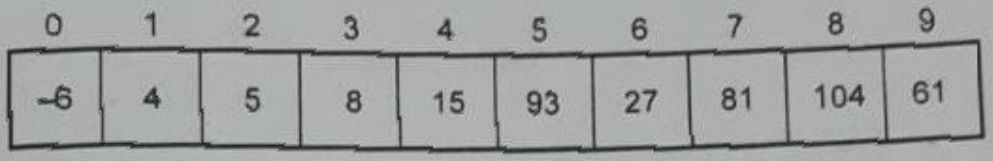
Code for sorting of list :

```
void SortList() {
    //Node current will point to head
    struct node *current = head, *new_temp = NULL;
    int temp;

    if(head == NULL) {
       return;
    }
    else {
       while(current != NULL) {
          //Node new_temp will point to node next to current
```

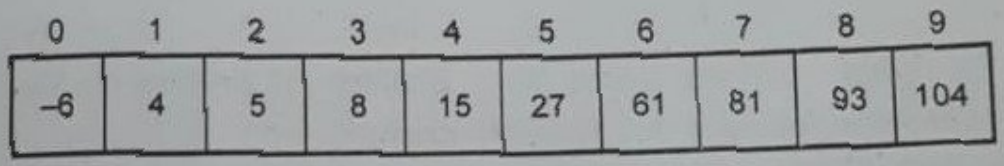
Pass 5 :



Swap 61 with 15. The array will be



Continue in this manner and finally we will get the sorted list as



Q.3 a) *What is linked list ? Write a pseudo C++ code to sort the elements.* **[9]**

Ans. : Linked list : Refer Q.2 of Chapter 2.

Code for sorting of list :

```
void SortList() {
    //Node current will point to head
    struct node *current = head, *new_temp = NULL;
    int temp;

    if(head == NULL) {
       return;
    }
    else {
       while(current != NULL) {
          //Node new_temp will point to node next to current
```

```
            new_temp = current->next;
            while(new_temp != NULL) {
          //If current node's data is greater than new_temp's node data,
          //swap the data between them
              if(current->data > new_temp->data) {
                temp = current->data;
                current->data = new_temp->data;
                new_temp->data = temp;
              }
              new_temp = new_temp->next;
            }
            current = current->next;
        }
    }
}
```

b) *What is doubly linked list ? Explain the process of deletion of an element from doubly linked list with example.*
(Refer Q.15, Q.17 of Chapter 4) [9]

OR

Q.4 a) *Explain generalized linked list with example.*
(Refer Q.27 of Chapter 4) [9]

b) *Write pseudo C++ code for addition of two polynomials using singly linked list.* **(Refer Q.26 of Chapter 4)** [9]

Q.5 a) *Write an algorithm for postfix evaluation with suitable example.* **(Refer Q.12 of Chapter 5)** [8]

b) *What is concept of recursion ? Explain the use of stack in recursion with example.* **(Refer Q.16, Q.18 of Chapter 5)** [9]

OR

Q.6 a) *What is need to convert the infix expression into postfix; convert the following expression into postfix expression*
*(a + b) * d + e/(f + a * d) + c.* [8]

Ans. : Need for conversion of infix to postfix :

1) Postfix expressions are easier for computers to evaluate because they do not require precedence and associativity rules to follow.

2) The postfix expressions are unambiguous. Hence unambiguous result can be obtained on evaluation of these expressions

3) Postfix expressions can be evaluated efficiently using stack. Stack is a simple data structure.

Let the infix expression be (a+b)*d+e/(f+a*d)+c

Input	Action	Stack	Postfix expression
(	Push	(	
a	Print a	(	a
+	Push	(+	a
b	Print b	(+	ab
)	Pop +, print, Pop (	empty	ab+
*	Push *	*	
d	Print d	*	ab+d
+	Pop *, print, Push +	+	ab+d*
e	Print e	+	ab+d*e
/	Push /	+/	ab+d*e
(	Push (	+/(	ab+d*e
f	Print f	+/(	ab+d*ef
+	Push +	+/(+	ab+d*ef
a	Print a	+/(+	ab+d*efa
*	Push *	+/(+*	ab+d*efa
d	Print d	+/(+*	ab+d*efad

)	Pop *, print Pop +, print Pop (	+/	ab+d*efad*+
+	Pop /, print, Pop +, print Push +	+	ab+d*efad*+/+
c	Print c	+	ab+d*efad*+/+c
End of Input	Pop +, print	empty	ab+d*efad*+/+c+

Postfix expression is : **ab+d*efad*+/+c+**

b) *What is backtracking algorithm design strategy ? How stack is useful in backtracking ?* **(Refer Q.21, Q.23 of Chapter 5)** [9]

Q.7 a) *Write pseudo C++ code to represent dequeue and perform the following operations on dequeue :*
i) Create ii) Insert iii) Delete iv) Display. [8]

Ans. :

```
/**********************************************************************
Program To implement Doubly ended queue using arrays
**********************************************************************/
#include<iostream>
#include<stdlib.h>
#define size 5
using namespace std;
class Dqueue
{
    private:
        int que[size];
    public:
        int front,rear;
        Dqueue();
```

```
        int Qfull();
        int Qempty();
        int insert_rear(int item);
        int delete_front();
        int insert_front(int item);
        int delete_rear();
        void display();
};

Dqueue::Dqueue()
{
    front=-1;
    rear=-1;
    for(int i=0;i<size;i++)
        que[i]=-1;

}

int Dqueue::Qfull()
{
    if(rear==size-1)
        return 1;
    else
        return 0;
}

int Dqueue::Qempty()
{
    if((front>rear)||(front==-1&&rear==-1))
        return 1;
    else
        return 0;
}
```

```
int Dqueue::insert_rear(int item)
{
    if(front==-1&&rear==-1)
        front++;
    que[++rear]=item;
    return rear;
}

int Dqueue::delete_front()
{
    int item;
    if(front==-1)
        front++;
    item=que[front];
    que[front]=-1;
    front++;
    return item;
}

int Dqueue::insert_front(int item)
{
    int i,j;
    if(front==-1)
        front++;
    i=front-1;
    while(i>=0)
{
        que[i+1]=que[i];
    i--;
}
j=rear;
while(j>=front)
{
```

```
    que[j+1]=que[j];
    j--;
  }
  rear++;
  que[front]=item;
  return front;
 }
int Dqueue::delete_rear()
 {
    int item;
    item=que[rear];
    que[rear]=-1;/*logical deletion*/
    rear--;
    return item;
}
void Dqueue::display()
{
    int i;
    cout<<"\n Straight Queue is:";
       for(i=front;i<=rear;i++)
    cout<<" "<<que[i];
}
int main()
{
    int choice,item;
    char ans;
    ans='y';
    Dqueue obj;
    cout<<"\n\t\t Program For doubly ended queue using arrays";
    do
    {
```

```
cout<<"\n1.insert by rear\n2.delete by front\n3.insert by
        front\n4.delete by rear";
cout<<"\n5.display\n6.exit";
cout<<"\n Enter Your choice ";
cin>>choice;
switch(choice)
{
case 1:if(obj.Qfull())
          cout<<"\n Doubly ended Queue is full";
       else
       {
       cout<<"\n Enter The item to be inserted";
       cin>>item;
       obj.rear=obj.insert_rear(item);
       }
     break;
case 2:if(obj.Qempty())
       cout<<"\n Doubly ended Queue is Empty";
       else
       {
       item=obj.delete_front();
       cout<<"\n The item deleted from queue is "<<item;
       }
       break;
case 3:if(obj.Qfull())
       cout<<"\n Doubly ended Queue is full";
       else
       {
       cout<<"\n Enter The item to be inserted";
       cin>>item;
       obj.front=obj.insert_front(item);
```

```
        }
        break;
case 4:if(obj.Qempty())
        cout<<"\n Doubly ended Queue is Empty";
        else
        {
        item=obj.delete_rear();
        cout<<"\n The item deleted from queue is "<<item;
        }
        break;
case 5:obj.display();
        break;
case 6:exit(0);
}
cout<<"\n Do You Want To Continue?";
cin>>ans;
}while(ans=='y'||ans=='Y');
return 0;
}
```

b) *What is circular queue ? Explain the advantages of circular queue over linear queue.* **(Refer Q.6 of Chapter 6)** [9]

Ans. : Advantages of circular queue over linear queue :

1) **Efficient memory utilization :** In a linear queue, if the queue is full and we try to enqueue an element, we will not be able to do so even if there are empty spaces at the front of the queue. This is because the front and rear pointers cannot be moved backwards. In a circular queue, however, the front and rear pointers can wrap around to the beginning of the queue, so we can always enqueue an element even if the queue is full.

2) Simple implementation : The implementation of a circular queue is simpler than the implementation of a linear queue. This is because the front and rear pointers in a circular queue can be implemented using a single integer variable, while the front and rear pointers in a linear queue require two integer variables.

OR

Q.8 a) *Define queue as an ADT. Write pseudo C++ code to represent queue.* **(Refer Q.3, Q.4 and Q.5 of Chapter 6)** [8]

b) *Explain array implementation of priority queue with all basic operations.* [9]

Ans. : C++ Program

```
/*****************************************************
Program for implementing the ascending priority Queue
*****************************************************/

    /*Header Files*/
    #include<iostream>
    #define SIZE 5
    using namespace std;
    class Pr_Q
    {
     private:
        int que[SIZE];
     public:
        int rear,front;
        Pr_Q();
        int insert(int rear,int front);
        int Qfull(int rear);
        int delet(int front);
        int Qempty(int rear,int front);
        void display(int rear,int front);
    };
    Pr_Q::Pr_Q()
```

```
{
 front=0;
 rear=-1;
}
int Pr_Q::insert(int rear,int front)
{
 int item,j;
 cout<<"\nEnter the element: ";
 cin>>item;
 if(front ==-1)
    front++;
 j=rear;
 while(j>=0 && item<que[j])
 {
  que[j+1]=que[j];
  j--;
 }
 que[j+1]=item;
 rear=rear+1;
 return rear;
}
int Pr_Q::Qfull(int rear)
{
 if(rear==SIZE-1)
  return 1;
 else
 return 0;
}
int Pr_Q::delet(int front)
{
```

```
cout<<"\n1.Insert\n2.Delete\n3.Display";
cout<<"\n Enter Your Choice: ";
cin>>choice;
switch(choice)
{
  case 1:if(obj.Qfull(obj.rear))
           cout<<"\n Queue IS full";
       else
         obj.rear=obj.insert(obj.rear,obj.front);
       break;
  case 2:if(obj.Qempty(obj.rear,obj.front))
           cout<<"\n Cannot delete element";
       else
          obj.front=obj.delet(obj.front);
       break;
  case 3:if(obj.Qempty(obj.rear,obj.front))
           cout<<"\n Queue is empty";
       else
       obj.display(obj.rear,obj.front);
       break;
default:cout<<"\n Wrong choice: ";
      break;
}
 cout<<"\n Do You Want TO continue?";
 cin>>ans;
}while(ans=='Y'||ans=='y');
return 0;
}
```