INDEX

6.2(a)	Write a program to multiply polynomials using linked list.	39-41		
6.2(b)	Write a program to implement basic operations (push, pop, display) of stack using array.	42-44		
7.1	Write a program to implement basic operations (push, pop, display) of stack using linked list.	45-46		
7.2(a)	Write a program which performs postfix evaluation.	47-49		
7.2(b)	Write a program to convert infix to postfix expression.	50-51		
7.3	Write a program to convert decimal number to octal number.	52-53		
8.1	Write a program to sort elements using quick sort using stack.	54-57		
8.2(a)	Write a program to find GCD of two numbers.	58		
8.2(b)	Write a program to sort elements using quick sort using stack.	59-60		
8.2(c1)	Write a program to find factorial of number using tail recursion.	61		
8.2(c2)	Write a program to find factorial of number using non-tail recursion.	62		
8.2(d1)	Write a program to find fibonacci series Using tail recursion.	63		
8.2(d2)	Write a program to find fibonacci series Using non-tail recursion.	64		
9.1	Write a program to implement basic operations (Insert,Delete and display) of queue using array .	65-67		

9.2	Write a program to implement basic operations (Insert,Delete and display) of queue using linked list.	68-70		
9.3(a)	Write a program to implement input restricted queue.	71-73		
9.3(b)	Write a program to implement output restricted queue.	74-76		
9.4(a)	Write a program to implement priority queue by array.	77-80		
9.4(b)	Write a program to implement priority queue by linked list.	81-83		
10.1	Write a program to implement insertion, deletion, height of tree in binary tree.	84-88		
10.2	Write a program to implement insertion, deletion, height of tree in binary search tree.	89-91		
11	Write a program to insertion, traversal, in AVL tree.	92-94		
12.1	Write a program to implement insertion, deletion, and sorting in Heap.	95-96		
12.2	Write a program to implement insertion, deletion of edge and display Graph.	97-98		
13.1(a)	Write a program to find path matrix using powers of matrix.	99-101		
13.1(b)	Write a program to find path matrix using Warshall algorithm.	102-104		
13.2(c)	Write a program to implement Breadth first and Depth first search.	105-109		
13.3	Write a program to find shortest path using Dijkstra algorithm.	110-112		

1. Write a program to implement

a) Insertion in an Array

b) deletion in an Array

c) Sorting elements of an array

d) Count the frequency of elements

```
#include<iostream>
#include<stdlib.h>
using namespace std;
void sorting(int*,int);
void insertion_beg(int *no,int *n)
{
	int i;
	for(i=(*n)-1;i>=0;i--)
	{
		no[i+1]=no[i];
	}
	cout<<"enter the no. u want to insert :\n";
	cin>>no[0];
	*n=(*n)+1;
}
void insertion_mid(int *no,int *n)
{
	int i,temp;
	cout<<"enter the index at which u want to insert :\n";
	cin>>temp;
	for(i=*(n)-1;i>=temp;i--)
	{
		no[i+1]=no[i];
	}
	cout<<"enter the no u want to insert :\n";
	cin>>no[temp];
	*n=(*n)+1;
}
void insertion_end(int *no,int *n)
{
	cout<<"enter the element u want to insert :\n";
	cin>>no[*n];
	*n=(*n)+1;
}
void delete_mid(int *no,int *n)
{
	int i,temp;
	cout<<"enter the index at which u want to delete :\n";
	cin>>temp;
	for(i=temp+1;i<*n;i++)
	{
		no[i-1]=no[i];
```

```
        }
        *n=(*n)-1;
}
void delete_element(int *no,int *n)    //deletion in array
{
        int temp,i,j;
        cout<<"enter the element u want to delete\n";
        cin>>temp;
        for(i=*n-1;i>=0;i--)
        {
                if(temp==no[i])
                {
                        for(j=i+1;j<*n;j++)
                        {
                                no[j-1]=no[j];
                        }
                        i--;
                        *n=(*n)-1;
                }
        }
}
void sorting(int *no,int n)             //sorting
{
        int i,j,c=0,temp;
        for(i=0;i<n;i++)
        {
                for(j=i+1;j<n;j++)
                {
                        if(no[j]<no[i])
                        {
                                temp=no[j];
                                no[j]=no[i];
                                no[i]=temp;
                                c++;
                        }
                }
        }
        cout<<"your array has been sorted in" <<c<<"steps\n";
}
void freq(int *no,int n,int x)                                  //frequency of element
{
        int i,c=0;
        for(i=0;i<n;i++)
        {
                if(no[i]==x)
                        c++;
        }
        cout<<x<<"appeared "<<c<<"times\n";
}
int main()
```

```
{
    int n,i;
    cout<<"enter the length of array: ";
    cin>>n;
    cout<<"enter the elements of array";
    int no[100],temp;
    for(i=0;i<n;i++)
    {
        cin>>no[i];
    }
    cout<<"\n\n1 for insertion at beg.\n2 for insertion    in mid\n3 for insertion at end\n";
    cout<<"4 for deletion at end\n5 for deletion at mid\n6 for deleting an element from complete
      array\n7 for sorting\n8 for print\n9 for frequency \n10for exit\n ";
    while(1)
    {
        int flag=0,c=0,lb,ub,c_b=0;
        cout<<"\nenter your choice :";
        int ch,j=0;
        cin>>ch;
        switch(ch)
        {
        case 1:
            insertion_beg(no,&n);
            break;
        case 2:
            insertion_mid(no,&n);
            break;
        case 3:
            insertion_end(no,&n);
            break;
        case 4:
            n=n-1;
            break;
        case 5:
            delete_mid(no,&n);
            break;
        case 6:
            delete_element(no,&n);
            break;
        case 7:
            sorting(no,n);
            break;
        case 8:
            print(no,n);
            break;
        case 9:
            cout<<"enter the no. u want to search: ";
            int x;
            cin>>x;
            freq(no,n,x);
```

```
            break;
        case 10:
            exit(0);
            break;
        default:
            cout<<"enter correct choice\n";
        }
    }
}
```

"F:\Ankit\array operations.exe"

```
enter the index at which u want to insert :
3
enter the no u want to insert :
26

enter your choice :8
10 20 30 26 40 50

enter your choice :5
enter the index at which u want to delete :
5

enter your choice :8
10 20 30 26 40

enter your choice :7
your array has been sorted in1steps

enter your choice :9
enter the no. u want to search: 10
10appeared 1times

enter your choice :8
10 20 26 30 40

enter your choice :10

Process returned 0 (0x0)   execution time : 65.805 s
Press any key to continue.
```

2.(a) Write a program to reverse an Array without using extra storage.

```
#include<iostream>
using namespace std;
int main()
{
    int n, i, a[10];
    cout<<"Enter total number of elements ";
    cin>>n;
    cout<<"enter elements of array ";
    for(i=0; i<n; i++)
    {
        cin>>a[i];
    }
    for(i=0;i<n/2;i++)
    {
        a[i]=a[i]+a[n-1-i];
        a[n-1-i]=a[i]-a[n-1-i];
        a[i]=a[i]-a[n-1-i];
    }
    cout<<"reversed array";
    for(i=0; i<n; i++)
    {
        cout<<a[i]<<" ";
    }
    return 0;
}
```

"F:\Ankit\reverse array without using extra space.exe"

```
Enter total number of elements 6
enter elements of array 2 6 4 8 5 9
reversed array9 5 8 4 6 2
Process returned 0 (0x0)   execution time : 23.208 s
Press any key to continue.
```

2.(b) Write a program to implement linear and binary search.

```
#include<iostream>
Using namespace std;
void binarysearch(int *no,int n)
{
	sorting(no,n);
	int temp,i,flag=0,c_b=0;
	cout<<"enter the no. u want to search :\n";
	cin>>temp;
	int lb=0,ub=n-1;
	while(ub>=lb)
	{
		if(temp>no[(ub+lb)/2])
		{
			lb=((ub+lb)/2)+1;
			c_b++;
		}
		else if(temp<no[(ub+lb)/2])
		{
			ub=((ub+lb)/2)-1;
			c_b++;
		}
		else if(temp==no[(ub+lb)/2])
		{
			c_b++;
			flag=1;
			break;
		}
	}
	if(flag==1)
		cout<<"yes"<<" couter="<<c_b<<"\n";
	else
		cout<<"no\n";

}
void linear_search(int *no,int n)
{
	int temp,i,j,flag;
	cout<<"enter the no. u want to search :\n";
	cin>>temp;
	for(i=0;i<n;i++)
	{
		if(temp==no[i])
		{
			flag=1;
			break;
		}
```

```
        }
        if(flag==1)
                cout<<"yes it is present \n";
            else
                cout<<"no it is not present\n" ;
}
int main()
{
        int n,i;
        cout<<"enter the length of array: ";
        cin>>n;
        cout<<"enter the elements of array";
        int no[100],temp;
        for(i=0;i<n;i++)
        {
            cin>>no[i];
        }
        int ch;
        cout<<" 1 for linear search \n 2 for binary search \n 3 for exit";
        while(1)
        {
            cout<<"\nenter your choice :";
            cin>>ch;
            switch(ch)
            {
            case 1:
                linear_search(no,n);
                break;
            case 2:
                binarysearch(no,n);
                break;
            case 3:
                exit(0);
            }
    }
}
```

"F:\Ankit\linear and binary search.exe"

```
enter the length of array: 7
enter the elements of array5 4 6 8 9 1 7
 1 for linear search
 2 for binary search
 3 for exit
enter your choice :2
enter the no. u want to search :
18
no

enter your choice :1
enter the no. u want to search :
4
yes it is present

enter your choice :3

Process returned 0 (0x0)   execution time : 53.039 s
Press any key to continue.
```

2.(c) Write a program for matrix multiplication.

```
#include<iostream>
using namespace std;
int main()
{
	int m,n;
	cout<<"enter order of mat1\n";
	cin>>m>>n;
	int q,p;
	cout<<"enter order of mat2\n";
	cin>>q>>p;
	if(n!=q)
		cout<<"matrix are not multiplicable\n";
	else
	{int a[m][n],b[n][p],c[m][p],i,j,k;
	cout<<"enter elements of mat1\n";
	for(i=0;i<m;i++)
		for(j=0;j<n;j++)
		cin>>a[i][j];
	cout<<"enter elements of mat 2\n";
	for(i=0;i<n;i++)
		for(j=0;j<p;j++)
		cin>>b[i][j];
	for(i=0;i<m;i++)
	{
		for(j=0;j<p;j++)
		{
			c[i][j]=0;
			for(k=0;k<n;k++)
			{
				c[i][j]=c[i][j]+(a[i][k]*b[k][j]);
			}
		}
	}
	cout<<"required matrix is:\n";
	for(i=0;i<m;i++)
		{for(j=0;j<p;j++)
		{cout<<c[i][j]<<" ";}
		cout<<"\n";}
	}
}
```

"F:\Ankit\matrix multiplication.exe"

```
enter order of mat1
3
3
enter order of mat2
3 2
enter elements of mat1
1 2 3 4 5 6 7 8 9
enter elements of mat 2
1 0 0
0 0 1
required matrix is:
1 3
4 6
7 9

Process returned 0 (0x0)   execution time : 41.865 s
Press any key to continue.
```

3.(a) Write a program to implement sparse matrix and transpose of matrix.

```
#include<iostream>
using namespace std;
struct sparse
{
	int r,c,v;
};
void getinput(sparse *s,int x)
{
	cout<<"enter the order of matrix :\n";
	int n,m;
	cin>>n>>m;
	s[0].r=n;s[0].c=m;s[0].v=x;
	cout<<"enter row ,column and value of matrix :\n";
	int i;
	for(i=1;i<=x;i++)
	{
		cin>>s[i].r>>s[i].c>>s[i].v;
	}
}
void transpose(sparse *s)
{
	int i;
	for(i=0;i<=s[0].v;i++)
	{
		int temp=s[i].r;
		s[i].r=s[i].c;
		s[i].c=temp;
	}
}
void print(sparse *s)
{
	int i,j,k;
	for(i=0;i<s[0].r;i++)
	{
		for(j=0;j<s[0].c;j++)
		{
		    int flag=0;
		      for(k=1;k<=s[0].v;k++)
		      {
			if(i==s[k].r && j==s[k].c)
			{
				flag=1;
				break;
			}
		      }
		      if(flag==1)
```

```
                cout<<s[k].v<<" ";
            else
                cout<<0<<" ";
        }
        cout<<"\n";
    }
}
int main()
{
    cout<<"enter the no. of non zero elements: ";
    int n;
    cin>>n;
    sparse s1[n+1];
    getinput(s1,n);
    cout<<"enter the no. of non zero elements: ";
    cin>>n;
    sparse s2[n+1];
    getinput(s2,n);
    cout<<"before transpose :\n";
    print(s1);
    transpose(s1);
    cout<<"after transpose :\n";
    cout<<"\n";
    print(s1);
}
```

"F:\Ankit\sparse matrix (2).exe"

```
enter the no. of non zero elements: 4
enter the order of matrix :
4 3
enter row ,column and value of matrix :
0 0 1
0 1 2
2 2 4
3 1 3
enter the no. of non zero elements: 2
enter the order of matrix :
3 2
enter row ,column and value of matrix :
0 0 1
2 1 2
before transpose :
1 2 0
0 0 0
0 0 4
0 3 0
after transpose :

1 0 0 0
2 0 0 3
0 0 4 0

Process returned 0 (0x0)   execution time : 47.468 s
Press any key to continue.
```

3.(b) Write a program for matrix addition using sparse.

```
#include<iostream>
using namespace std;
main()
{
	int m1,n1,t1,t2;
	cout<<"enter order of matrix 1\n";
	cin>>m1>>n1;
	cout<<"enter no. of non zero elements in matrix 1\n";
	cin>>t1;
	int mat1[t1][3];
	cout<<"enter row, column, value of each element for mat 1\n";
	for(int i=0;i<t1;i++)
		for(int j=0;j<3;j++)
			cin>>mat1[i][j];
	cout<<"\n\n\n";
	cout<<"enter no. of non zero elements for mat 2\n";
	cin>>t2;
	int mat2[t2][3];
	cout<<"enter row, column, value of each element for mat 2\n";
	for(int i=0;i<t2;i++)
		for(int j=0;j<3;j++)
			cin>>mat2[i][j];
	cout<<"\n\n\naddition of matrices is:\n";
	int mat[m1][n1];
	for(int i=0;i<m1;i++)
	{
		for(int j=0;j<n1;j++)
		{
			mat[i][j]=0;
			for(int k=0;k<t1;k++)
			{
				if(mat1[k][0]==i&&mat1[k][1]==j)
				{
					mat[i][j]+=mat1[k][2];
					break;
				}
			}
			for(int k=0;k<t2;k++)
			{
				if(mat2[k][0]==i&&mat2[k][1]==j)
				{
					mat[i][j]+=mat2[k][2];
					break;
				}
			}
			cout<<mat[i][j]<<"    ";
		}
```

```
        cout<<"\n";
    }
}
```

"F:\Ankit\matrix addition using sparse.exe"

```
enter order of matrix 1
3 3
enter no. of non zero elements in matrix 1
3
enter row, column, value of each element for mat 1
0 0 1
1 1 2
2 2 3



enter no. of non zero elements for mat 2
3
enter row, column, value of each element for mat 2
0 0 1
0 1 2
0 2 4



addition of matrices is:
2  2  4
0  2  0
0  0  3

Process returned 0 (0x0)   execution time : 39.977 s
Press any key to continue.
```

3.(c) Write a program to multiply sparse matrices.

```
#include<iostream>
using namespace std;
struct sparse
{
	int r,c,v;
};
void getinput(sparse *s,int x)
{
	cout<<"enter the order of matrix :\n";
	int n,m;
	cin>>n>>m;
	s[0].r=n;s[0].c=m;s[0].v=x;
	cout<<"enter row ,column and value of matrix :\n";
	int i;
	for(i=1;i<=x;i++)
	{
		cin>>s[i].r>>s[i].c>>s[i].v;
	}
}
void transpose(sparse *s)
{
	int i;
	for(i=0;i<=s[0].v;i++)
	{
		int temp=s[i].r;
		s[i].r=s[i].c;
		s[i].c=temp;
	}
}
void print(sparse *s)
{
	int i,j,k;
	for(i=0;i<s[0].r;i++)
	{
		for(j=0;j<s[0].c;j++)
		{
			int flag=0;
			  for(k=1;k<=s[0].v;k++)
			  {
				if(i==s[k].r && j==s[k].c)
				{
					flag=1;
					break;
				}
			  }
			  if(flag==1)
				cout<<s[k].v<<" ";
```

```
                else
                        cout<<0<<" ";
            }
            cout<<"\n";
        }
}
void mult(sparse *m1,sparse *m2,sparse *m3)
{
        transpose(m2);
        int c=0;
        m3[0].r=m1[0].r;
        m3[0].c=m2[0].r;
        int i,j,k,l,sum;
        for(i=0;i<m1[0].r;i++)
        {
            for(j=0;j<m2[0].r;j++)
            {
                sum=0;
                for(k=1;k<=m1[0].v;k++)
                {
                        if(m1[k].r==i)
                        {
                                for(l=1;l<=m2[0].v;l++)
                                {
                                    if(m2[l].r==j)
                                    {
                                            if(m2[l].c==m1[k].c)
                                            {
                                                    sum=sum+((m1[k].v)*(m2[l].v));
                                            }
                                    }
                                }
                        }
                }
                if(sum!=0)
                {
                c++;
                m3[c].r=i;
                m3[c].c=j;
                m3[c].v=sum;
                }
            }
        }
        m3[0].v=c;
}
int main()
{
        cout<<"enter the no. of non zero elements: ";
        int n;
        cin>>n;
```

```
    sparse s1[n+1];
    getinput(s1,n);
    cout<<"enter the no. of non zero elements: ";
    cin>>n;
    sparse s2[n+1];
    getinput(s2,n);
    sparse s3[(s1[0].r)*(s2[0].c)+1];
    mult(s1,s2,s3);
    cout<<"\nafter multiplication matrix is :\n";
    print(s3);

}
```

"F:\Ankit\multiply sparse matrices.exe"

```
enter the no. of non zero elements: 3
enter the order of matrix :
3 3
enter row ,column and value of matrix :
0 0 1
1 1 3
2 2 2
enter the no. of non zero elements: 4
enter the order of matrix :
3 3
enter row ,column and value of matrix :
0 2 4
1 2 2
2 0 1
2 2 1

after multiplication matrix is :
0 0 4
0 0 6
2 0 2

Process returned 0 (0x0)   execution time : 46.478 s
Press any key to continue.
```

4. Write a program to implement

a) **Create and Traverse a Singly linked list**
b) **Double of original value of Linked List**
c) **Sum of previous elements of linked list**
d) **Find Min and Max element from integer Linked List**
e) **Insertion in a Singly Linked List at Beginning, middle and end position**
f) **Deletion of node from beginning , middle and end of list**

```
#include<iostream>
#include<stdlib.h>
using namespace std;
struct node
{
	int data;
	node *next;
};
node* create(int n)
{
	node *temp=new node;
	temp->next=NULL;
	temp->data=n;
	return temp;
}
void insertion(node **head,int n)		//insertion
{
	node *temp=*head;
	if(temp==NULL)
	{
		*head=new node;
		(*head)->data=n;
		(*head)->next=NULL;
	}
	else
	{
		while((temp)->next!=NULL)
		{

			(temp)=(temp)->next;
		}
		(temp)->next=new node;
			(temp)->next->next=NULL;
			(temp)->next->data=n;
	}
}


void print(node *temp)
```

```
{
        while(temp!=NULL)
        {
                cout<<temp->data<<"\n";
                temp=temp->next;
        }
}
void double_value(node *temp)
{
         while(temp!=NULL)
        {
                temp->data=temp->data*2;
                temp=temp->next;
        }
}
void sum(node *temp)
{
        int sum=0;
         while(temp!=NULL)
        {
                sum=temp->data+sum;
                temp->data=sum;
                temp=temp->next;
        }
}
void sum_last(node *temp)
{
        int sum,x;
        while(temp!=NULL)
        {
                x=temp->data;
                temp->data=x+sum;
                sum=x;
                temp=temp->next;
        }
}
int largest(node *temp)                 //largest
{
        int largest=temp->data;
        while(temp!=NULL)
        {
                if(largest<temp->data)
                {
                        largest=temp->data;
                }
                temp=temp->next;
        }
        return largest;
}
int smallest(node *temp)
```

```
{
	int smallest=temp->data;
	while(temp!=NULL)
	{
		if(lsmallest>temp->data)
		{
			smallest=temp->data;
		}
		temp=temp->next;
	}
	return smallest;
}
void deletion_begin(node **head)		//deletion at begin
{
	node *temp=*head;
	if(temp==NULL)
		cout<<"list is already empty.";
	else
	{
		temp=temp->next;
		delete head;
		*head=temp;
	}
}
void deletion_end(node *head)		//deletion at end
{
	node *temp,*t;
	temp=head;
	if(temp==NULL)
		cout<<"List is already empty";
	else
	{
		while(temp->next!=NULL)
		{
			t=temp->next;
			if(t->next==NULL)
				break;
			else
				temp=temp->next;
		}
		temp->next=NULL;
		delete t;
	}

}
void deletion_mid(node *head)		//deletion in mid
{
	node *temp,*t;
	int n;
	cout<<"enter the data you want to delete";
```

```
        cin>>n;
        temp=head;
        if(temp==NULL)
                cout<<"list is empty";
        else
        {
                t=temp->next;
                cout<<"\n\n\n"<<t->next;
                while(t->next!=NULL)
                {
                        t=temp->next;
                        if(t->data!=n)
                                temp=temp->next;
                        else
                        {
                                temp->next=t->next;
                                break;
        }
        }
        if(t->data==n && t->next==NULL)
                cout<<"\nend of list\n";
        else if(t->next==NULL)
        cout<<"\nwrong input\n";
        }
}
int main()
{
        node *head=NULL;
        int ch,x,y;
        cout<<"1 insertion \n2 largest\n3 least elmenent\n4double each value\n5 previous elements
       sum\n6 sum only last one\n7 print\n8 deletion at beg\n;
        cout<<"9 deletion_mid \n10 deletion end\n11 exit\n";
        while(1)
        {
                cout<<"enter your choice :";
                cin>>ch;
                switch(ch)
                {
                case 1:
                        cout<<"enter data :";
                        cin>>x;
                        insertion(&head,x);
                        break;
                case 2:
                        y=largest(head);
                        cout<<y<<"\n";
                        break;
                case 3:
                        y=smallest(head);
                        cout<<y<<"\n";
```

```
            break;
        case 4:
            double_value(head);
            break;
        case 5:
            sum(head);
            break;
        case 6:
            sum_last(head);
            break;
        case 7:
            print(head);
            break;
        case 8:
            deletion_begin(&head);
            break;
        case 9:
            deletion_mid(head);
        case 10:
            deletion_end(head);
        case 11:
            exit(0);
            break;

        default:
            cout<<"enter correct choice :";
        }
    }
}
```

"F:\Ankit\linked list.exe"

```
1 insertion
2 largest
3 least elmenent
4double each value
5 previous elements     sum
6 sum only last one
7 print
8 deletion at beg
9 deletion_mid
10 deletion end
11 exit
enter your choice :1
enter data :25
enter your choice :1
enter data :36
enter your choice :1
enter data :96
enter your choice :1
enter data :46
enter your choice :2
96
enter your choice :7
25
36
96
46
enter your choice :8
enter your choice :7
36
96
46
enter your choice :5
enter your choice :7
36
132
178
enter your choice :11

Process returned 0 (0x0)   execution time : 47.400 s
Press any key to continue.
```

5. Write a program to implement

a) Frequency of elements in sorted and unsorted linked list

b) Swapping of 2 nodes in a linked list for consecutive and non-consecutive nodes

c) Reverse a Singly Linked List

d) Concatenation of two Linked list

e) Splitting of linked list in even and odd elements list

```
#include<iostream>
#include<stdlib.h>
using namespace std;
struct node
{
	int data;
	node *next;
};
void insertion(node **,int );
node* create(int n)
{
	node *temp=new node;
	temp->next=NULL;
	temp->data=n;
	return temp;
}
node *createlist()
{
	int i,n,x;
	cout<<"enter the length of 1 list :";
	cin>>n;
	cout<<"enter the data :";
	cin>>x;
	node *temp1=create(x);
	for(i=1;i<n;i++)
	{
		cin>>x;
		insertion(&temp1,x);
	}
	return temp1;
}
void insertion(node **head,int n)		//insertion of node
{
	node *temp=*head;
	if(temp==NULL)
	{
		*head=new node;
		(*head)->data=n;
		(*head)->next=NULL;
	}
	else
```

```
        {
            while((temp)->next!=NULL)
            {
                (temp)=(temp)->next;
            }
            (temp)->next=new node;
                (temp)->next->next=NULL;
                (temp)->next->data=n;
        }
}
void print(node *temp)
{
        while(temp!=NULL)
        {
            cout<<temp->data<<" ";
            temp=temp->next;
        }
        cout<<"\n";
}
void freq_sorted(node *temp)        //freq in sorted list
{
        int c=1;
        while(temp->next!=NULL)
        {
            int x=temp->data;
            if(temp->data==temp->next->data)
            {
                c++;
            }
            else
            {
                cout<<temp->data<<"-"<<c<<"\n";
                c=1;
            }
            temp=temp->next;
            if(temp->next==NULL)
            {
                cout<<temp->data<<"-"<<c<<"\n";
            }
        }
}
void freq_unsorted(node *head)          //freq in unsorted list
{
        node *temp1=head,*temp2;
        while(temp1!=NULL)
        {
            temp2=temp1->next;
            int c=1;
            while(temp2!=NULL)
            {
```

```
                if(temp1->data==temp2->data)
                {
                        c++;
                        temp2->data=temp2->next->data;
                        temp2->next=temp2->next->next;
                }
                else
                  temp2=temp2->next;
            }
            cout<<temp1->data<<" - "<<c<<"\n";
            temp1=temp1->next;
        }
}
void reverselist(node **head)
{
        node *temp=*head,*t;
          while(temp->next!=NULL)
        {
            t=temp->next;
            temp->next=t->next;
            t->next=*head;
            *head=t;
        }
}
node* concatenate()
{
      node *temp1=createlist(),*temp2=createlist();
        node *t=temp1;
        while(temp1->next!=NULL)
        {
            temp1=temp1->next;
        }
        temp1->next=temp2;
        return t;
}
node* swapnode(node *start,int x)
{
        int i;
        node *temp1,*temp,*temp2;
        temp=start;
        if(x>=2)
        {
            for(i=0;i<x-2;i++)
            {
                temp=temp->next;
            }
            temp2=temp;
            temp=temp->next;
            temp1=temp;
            temp=temp->next;
```

```
            temp1->next=temp->next;
            temp->next=temp1;
            temp2->next=temp;
        }
        else
        {
            start=temp->next;
            temp->next=start->next;
            start->next=temp;
        }
        return start;
    }
    node* swap_nc(node *start,int x,int y)      //swaping of nodes
    {
        int i;
        node *temp,*temp1,*temp2,*temp3;
        temp=start;
        if(x>=2)
        {
            for(i=0;i<x-2;i++)
                temp=temp->next;
            temp2=temp->next->next;
            temp1=temp2;
            for(i=0;i<y-x-2;i++)
                temp1=temp1->next;
            temp3=temp1->next;
            temp->next->next=temp3->next;
            temp1->next=temp->next;
            temp->next=temp3;
            temp3->next=temp2;
        }
        else
        {
            temp1=temp;
            temp2=temp->next;
            for(i=0;i<y-2;i++)
                temp1=temp1->next;
            start=temp1->next;
            temp1->next=temp;
            temp->next=start->next;
            start->next=temp2;
        }
        return start;
    }
    void splitting(node *head)
    {
        node *temp=head,*temp1=head->next,*temp3;
        while(temp!=NULL && temp->next!=NULL && temp->next->next!=NULL)
        {
            cout<<temp->data<<" ";
```

```
        temp=temp->next->next;
    }
    if(temp!=NULL)
        cout<<temp->data<<" ";
    while(temp1!=NULL && temp1->next!=NULL && temp1->next->next!=NULL)
    {
        cout<<temp1->data<<" ";
        temp1=temp1->next->next;
    }
    if(temp1!=NULL)
        cout<<temp1->data<<" ";
    cout<<"\n";
}


int main()
{
    cout<<" 1 for frequency in sorted \n 2 for frequency in unsorted \n 3 for reversing\n 4 for
concatenation \n 5 for print\n 6 for swapping consecutive \n ";
    cout<<"7 for swapping non consecutive \n 8 splitting\n";
    node *head=NULL;
    while(1)
    {
        cout<<"enter ur choice: ";
        int ch;
        cin>>ch;
        switch(ch)
        {
        case 1:
            head=createlist();
            freq_sorted(head);
            break;
        case 2:
            head=createlist();
            freq_unsorted(head);
            break;
        case 3:
            head=createlist();
            reverselist(&head);
            break;
        case 4:
            head=concatenate();
            break;
        case 5:
            print(head);
            break;
        case 6:
            swapnode(head,2);
            break;
        case 7:
```

```
            swap_nc(head,2,4);
            break;
        case 8:
            splitting(head);
            break;
        case 9:
            exit(0);
        }
    }
}
```

F:\Ankit\concatenation,swapping,splitting.exe

```
 1 for frequency in sorted
 2 for frequency in unsorted
 3 for reversing
 4 for concatenation
 5 for print
 6 for swapping consecutive
 7 for swapping non consecutive
 8 splitting
enter ur choice: 1
enter the length of 1 list :7
enter the data :1 1 1 2 3 3 4
1-3
2-1
3-2
4-1
enter ur choice: 3
enter the length of 1 list :5
enter the data :1 2 3 4 5
enter ur choice: 5
5 4 3 2 1
enter ur choice: 4
enter the length of 1 list :3
enter the data :4 5 6
enter the length of 1 list :5
enter the data :7 8 9 2 3
enter ur choice: 7
enter ur choice: 5
4 7 6 5 8 9 2 3
enter ur choice: 8
4 6 8 2 7 5 9 3
enter ur choice: 9

Process returned 0 (0x0)   execution time : 85.664 s
Press any key to continue.
```

6.1 Write a program to create doubly linked list which can store integers and write funcions to perform

a) Insertion in list
b) Deletion in list
c) Display of list in reverse direction
d) Display of list in forward direction

```
#include<iostream>
using namespace std;
struct node
{
	int data;
	node *next,*prev;
};
class linklist
{
private:
	node *head,*tail;
public:
	linklist()
	{
		head=NULL;
		tail=NULL;
	}
	void insertion(int j,int n)		//insertion of nodes
	{
		int i;
		if(head==NULL && tail==NULL)
		{
			head=new node;
			tail=head;
			head->data=n;
			head->next=NULL;
			head->prev=NULL;
			//cout<<1;
		}
		else
		{
			node *temp=head;
			for(i=1;i<j;i++)
			{
				if(temp->next==NULL)
				{
					break;
				}
				temp=temp->next;
			}
			if(temp->next!=NULL)
```

```
            {
            temp->next->prev=new node;
            temp->next->prev->data=n;
            temp->next->prev->prev=temp;
            temp->next->prev->next=temp->next;
            temp->next=temp->next->prev;
            //cout<<23;
            }
            else
            {
                temp->next=new node;
                temp->next->data=n;
                temp->next->next=NULL;
                temp->next->prev=temp;
                tail=temp->next;
            }
        }

}
void del(int i)        //deletion of nodes
{
        node *temp=head;
        if(i==1)
        {
            if(temp->next==NULL)
            {
                head=NULL;
                tail=NULL;
                delete temp;
            }
            else
            {
                temp->next->prev=NULL;
                head=temp->next;
                delete temp;

            }
        }
        else
        {
            int j;
            for(j=1;j<i;j++)
            {
                temp=temp->next;
            }
            if(temp->next==NULL)
            {
                temp->prev->next=NULL;
                tail=temp->prev;
                delete temp;
```

```
                }
                else
                {
                        temp->prev->next=temp->next;
                        temp->next->prev=temp->prev;
                        delete temp;
                }
        }
    }
    void print()
    {
        cout<<"forward order :\n";
        node *temp=head;
        while(temp!=NULL)
        {
            cout<<temp->data<<" ";
            temp=temp->next;
        }
        cout<<"\n";
    }
    void rev_print()
    {
        cout<<"reverse order :\n";
        node *temp=tail;
        while(temp!=NULL)
        {
            cout<<temp->data<<" ";
            temp=temp->prev;
        }
    }
};
int main()
{
    linklist l;
    int i,n;
    cout<<"enter the index after which u want to insert and data :";
    cin>>i>>n;
    l.insertion(i,n);
    cin>>i>>n;
    l.insertion(i,n);
    cin>>i>>n;
    l.insertion(i,n);
    cin>>i>>n;
    l.insertion(i,n);
    l.del(1);
    l.del(2);
    l.print();
    l.rev_print();

}
```

"F:\Ankit\doubly linked list.exe"

```
enter the index after which u want to insert and data :0 12
1 15
2 36
3 89
forward order :
15 89
reverse order :
89 15
Process returned 0 (0x0)   execution time : 26.544 s
Press any key to continue.
```

6.2 (a) WAP to create polynomial using linked list and perform addition and multiplication .

```
#include<iostream>
using namespace std;
struct node
{
	int data,power;
	node *next;
};
class linklist
{
private:
	node *head=NULL;
public:
	void insertion_end(int n,int x)
	{
		node *temp=head;
		if(temp==NULL)
		{
			head=new node;
			head->next=NULL;
			head->data=n;
			head->power=x;
		}
		else
		{
			while(temp->next!=NULL)
			{
				temp=temp->next;
			}
			temp->next=new node;
			temp->next->next=NULL;
			temp->next->data=n;
			temp->next->power=x;
		}
	}
	void print()
	{
		node *temp=head;
		if(temp!=NULL)
		{
			while(temp!=NULL)
			{
				cout<<temp->data<<"x^"<<temp->power<<"+";
				temp=temp->next;
			}
		}
	}
	node* return_head()
```

```
	{
		return head;
	}
	node** return_add()
	{
		return &head;
	}
	void sorted(node *list1,node *list2)
	{
			while(list1!=NULL || list2!=NULL)
			{
				if(list1!=NULL && list2!=NULL)
				{
					if(list1->power>list2->power)
					{
						insertion_end(list1->data,list1->power);
						list1=list1->next;
					}
					else if(list1->power<list2->power)
					{
						insertion_end(list2->data,list2->power);
						list2=list2->next;
					}
					else
					{
						insertion_end(list2->data+list1->data,list2->power);
						list2=list2->next;
						list1=list1->next;
					}

				}
				else if(list1==NULL && list2!=NULL)
				{
					insertion_end(list2->data,list2->power);
					list2=list2->next;
				}
				else if(list2 ==NULL && list1!=NULL)
				{
					insertion_end(list1->data,list1->power);
					list1=list1->next;
				}
			}
	}

};

int main()
{
	linklist l1,l2,l3;
	int i,n,x;
```

```
    cout<<"enter polynomial :\n";
    for(i=0;i<3;i++)
    {     cin>>n>>x;
          l1.insertion_end(n,x);
    }
     cout<<"enter polynomial :\n";
    for(i=0;i<2;i++)
    {
          cin>>n>>x;
          l2.insertion_end(n,x);
    }
        // l1.print();
          l3.sorted(l1.return_head(),l2.return_head());
          l3.print();


}
```

F:\Ankit\polynomial.exe

```
enter polynomial :
1 2
2 1
1 0
enter polynomial :
1 1
1 0
1x^2+3x^1+2x^0+
Process returned 0 (0x0)   execution time : 27.173 s
Press any key to continue.
```

6.2 (b) Write a program for polynomial multiplication.

```
#include<iostream>
using namespace std;
struct node
{
	int data,power;
	node *next;
};
class linklist
{
private:
	node *head=NULL;
public:
	void insertion_end(int n,int x)
	{
		node *temp=head;
		if(temp==NULL)
		{
			head=new node;
			head->next=NULL;
			head->data=n;
			head->power=x;
		}
		else
		{
			while(temp->next!=NULL)
			{
				temp=temp->next;
			}
			temp->next=new node;
			temp->next->next=NULL;
			temp->next->data=n;
			temp->next->power=x;
		}
	}
	void print()
	{
		node *temp=head;
		if(temp!=NULL)
		{
			while(temp!=NULL)
			{
				cout<<temp->data<<"x^"<<temp->power<<"+";
				temp=temp->next;
			}
		}
	}
	node* return_head()
	{
```

```
        return head;
    }
    node** return_add()
    {
        return &head;
    }
    void sorted(node *list1,node *list2)
    {
        node *temp1=list1,*temp2;
        while(temp1!=NULL)
        {
            temp2=list2;
            while(temp2!=NULL)
            {
                insertion_end((temp1->data)*(temp2->data),(temp1->power)+(temp2->power));
                temp2=temp2->next;
            }
            temp1=temp1->next;
        }
    }
    void mul(node *l3)
    {
        int sum=0;
        while(l3!=NULL)
        {
            sum=l3->data;
            node *temp=l3->next,*temp1=l3;
            while(temp!=NULL)
            {
                if(l3->power==temp->power)
                {
                    sum=sum+temp->data;
                    temp1->next=temp->next;
                    temp=temp->next;
                }
                else
                {
                      temp=temp->next;
                    temp1=temp1->next;
                }

            }
            insertion_end(sum,l3->power);
            l3=l3->next;
        }
    }

};

int main()
```

```
{
    linklist l1,l2,l3,l4;
    int i,n,x;
    for(i=0;i<3;i++)
    {   cin>>n>>x;
        l1.insertion_end(n,x);
    }
    for(i=0;i<3;i++)
    {
        cin>>n>>x;
        l2.insertion_end(n,x);
    }
      // l1.print();
        l3.sorted(l1.return_head(),l2.return_head());
        //l3.print();
        l4.mul(l3.return_head());
        l4.print()
}
```

"F:\Ankit\polynomial multiplication.exe"

```
enter polynomial :
1 2
2 1
1 0
enter polynomial :
1 2
2 1
1 0
1x^4+4x^3+6x^2+4x^1+1x^0+
Process returned 0 (0x0)   execution time : 32.841 s
Press any key to continue.
```

7.1 WAP a program to implement operations of stack .

```
#include<iostream>
#include<stdlib.h>
using namespace std;
struct arraystack
{
	int top,cap,base;
	long long int *no;
};
arraystack* create(int n)
{
	arraystack *head=(arraystack*)malloc(sizeof(arraystack));
	head->top=-1;
	head->cap=n;
	head->base=0;
	head->no=(long long int*)malloc(sizeof(long long int)*n);
	return head;
}
int full(arraystack *head)
{
	if(head->top==(head->cap)-1)
		return 1;
	return 0;

}
int blank(arraystack *head)
{
	if(head->top==-1)
		return 1;
	return 0;
}
void push(arraystack *head,long long int n)
{
	if(!full(head))
	{
		head->top++;
		head->no[head->top]=n;
	}
}
long long int pop(arraystack *head)
{
	int n;

	if(head->top!=-1)
	{
		n=head->no[head->top];
		head->top--;
		return n;
```

```
        }
}
int main()
{
        long long int n,k,i,x,greatest =0;
        cout<<"enter capacity of stack :\n";
        cin>>n;
        cout<<"enter elements\n";
        arraystack *Stack=create(n),*Stack1=create(n);
      for(i=0;i<n;i++)
        {
              cin>>x;
            push(Stack,x);
        }
        for(i=0;i<2;i++)
        {
              cout<<"popped element is :"<<pop(Stack)<<"\n";
        }



}
```

"F:\Ankit\operations of stack.exe"

```
enter capacity of stack :
5
enter elements
3 6 9 15 5
popped element is :5
popped element is :15

Process returned 0 (0x0)   execution time : 28.370 s
Press any key to continue.
```

7.2 (a) Write a program to convert Infix expression to postfix expression.

```
#include<iostream>
using namespace std;
struct stack1
{
	int top,size1;
	char *ptr;
};
stack1* create_stack(int k)
{
	stack1 *n;
	n=new stack1;
	n->top=-1;
	n->size1=k;
	n->ptr=new char[k];
	return n;
}
void push(char k,stack1 *n)
{
		n->top++;
		n->ptr[n->top]=k;
}
char pop(stack1 *n)
{
		n->top--;
		return n->ptr[n->top+1];
}
int main()
{
	stack1 *s;
	char a[100],b[100]={'\0'},c;
	s=create_stack(100);
	cout<<"Enter infix string\n";
	cin>>a;
	int i=0,j=0;
	while(a[i]!='\0')
		i++;
	a[i]=')';
	push('(',s);
	i=0;
	j=0;
	while(s->top!=-1)
	{
		if(a[i]>=48&&a[i]<=57)
		{
			b[j]=a[i];
			j++;
```

```
}
else if(a[i]!='('&&a[i]!=')')
{
        if(a[i]=='^')
            push('^',s);
        else if(a[i]=='/'||a[i]=='*')
        {
            abc: c=pop(s);
             if(c=='^')
             {
                    b[j]=c;
                    j++;
                    goto abc;
             }
             else
                    push(c,s);
             push(a[i],s);
        }
        else if(a[i]=='+'||a[i]=='-')
        {
            bcd: c=pop(s);
             if(c=='^'||c=='*'||c=='/')
             {
                    b[j]=c;
                    j++;
                    goto bcd;
             }
             else
                    push(c,s);
             push(a[i],s);
        }

}
else if(a[i]=='(')
        push(a[i],s);
else if(a[i]==')')
{
        while(1)
        {
             c=pop(s);
             if(c=='(')
                    break;
             else
             {
                    b[j]=c;
                    j++;
             }
        }
```

```
        }
        i++;
    }
    cout<<b<<"\n";
}
```

"F:\Ankit\Infix expression to postfix expression.exe"

```
Enter infix string
(2*(4+5)/7)
245+7/*

Process returned 0 (0x0)   execution time : 30.142 s
Press any key to continue.
```

7.2 (b)Write a program to evaluate postfix expression.

```
#include<iostream>
#include<malloc.h>
#include<stdio.h>
using namespace std;
struct arrstack
{
	int top;
	unsigned cap;
	int *ptr;
};
arrstack *temp;
void create_stack()
{
	temp=(arrstack *)malloc(sizeof(arrstack*));
	temp->top=0;
	temp->cap=20;
	temp->ptr=(int *)malloc(sizeof(int)*temp->cap);
}
void push(int x)
{
	temp->ptr[temp->top]=x;
	temp->top++;
}
int pop()
{
	int t;
		t=temp->ptr[temp->top-1];
		temp->top--;
		return t;
}
main()
{
	int i=0,x,y,value,t;
	create_stack();
	char s[50];
	cout<<"enter a string\n";
	gets(s);
	for(i=0;s[i]!='\0';i++)
	{
		if(s[i]>=48&&s[i]<=57)
		{
			t=0;
			while(s[i]!=',')
			{
				t=(t*10)+s[i]-48;
				i++;
			}
```

```
            push(t);
        }
        else if(s[i]=='*')
        {
            x=pop();
            y=pop();
            t=y*x;
            push(t);
        }
        else if(s[i]=='+')
        {
            x=pop();
            y=pop();
            t=y+x;
            push(t);
        }
        else if(s[i]=='-')
        {
            x=pop();
            y=pop();
            t=y-x;
            push(t);
        }
        else if(s[i]=='/')
        {
            x=pop();
            y=pop();
            t=y/x;
            push(t);
        }
    }
    value=pop();
    cout<<value;
}
```

"F:\Ankit\evaluate postfix expression.exe"

```
enter a string
2,3,5,+,7,/,*
2
Process returned 0 (0x0)   execution time : 29.654 s
Press any key to continue.
```

7.3 Write a program to convert decimal to octal using stack.

```
#include<iostream>
#include<stdlib.h>
using namespace std;
struct arraystack
{
	int top,cap;
	long long int *no;
};
arraystack* create(int n)
{
	arraystack *head=(arraystack*)malloc(sizeof(arraystack));
	head->top=-1;
	head->cap=n;
	// head->base=0;
	head->no=(long long int*)malloc(sizeof(long long int)*n);
	return head;
}
int full(arraystack *head)
{
	if(head->top==(head->cap)-1)
		return 1;
	return 0;

}
int blank(arraystack *head)
{
	if(head->top==-1)
		return 1;
	return 0;
}
void push(arraystack *head,long long int n)
{
	if(!full(head))
	{
		head->top++;
		head->no[head->top]=n;
	}
}
long long int pop(arraystack *head)
{
	int n;
	if(!blank(head))
	{
		n=head->no[head->top];
		head->top--;
		return n;
```

```
        }
}
void print(arraystack *head)
{
        while(!blank(head))
        {
                cout<<pop(head);
        }
}

int main()
{
        arraystack *s=create(100);
        long long int n;
        cout<<"enter the decimal form :\n";
        cin>>n;
        while(n>0)
        {
                push(s,n%8);
                n=n/8;
        }
        print(s);
}
```

"F:\Ankit\decimal to octal.exe"

```
enter the decimal form :
15
17
Process returned 0 (0x0)   execution time : 11.857 s
Press any key to continue.
```

8.1 Write a program to sort elements using quick sort using Stack.

```
#include<iostream>
using namespace std;
struct stack1
{
	int top,size1;
	int *ptr;
};
stack1* create_stack(int k)
{
	stack1 *n;
	n=new stack1;
	n->top=-1;
	n->size1=k;
	n->ptr=new int[k];
	return n;
}
int isfull(stack1 *n)
{
	if(n->top==n->size1-1)
		return 1;
	else
		return 0;
}
int isempty(stack1 *n)
{
	if(n->top==-1)
		return 1;
	else
		return 0;
}
void push(int k,stack1 *n)
{
	if(!isfull(n))
	{
		n->top++;
		n->ptr[n->top]=k;
	}
}
int pop(stack1 *n)
{
	if(!isempty(n))
	{
		n->top--;
		return n->ptr[n->top+1];
	}
```

```
}
inline void swap1(int &a,int &b)
{
	int c;
	c=a;
	a=b;
	b=c;
}
void display(stack1 *n)
{
	int a,b=0;
	a=n->top;
	while(b!=a+1)
	{
		cout<<n->ptr[b]<<" ";
		b++;
	}
	cout<<"\n";
}
int main()
{
	int n,j=0;
	stack1 *low,*up;
	low=create_stack(100);
	up=create_stack(100);
	cout<<"enter number of elements\n";
	cin>>n;
	int a[n];
	cout<<"enter values\n";
	while(j!=n)
	{
		cin>>a[j];
		j++;
	}
	push(0,low);
	push(n-1,up);
	int beg,en,left,right,loc,k;
	while(low->top!=-1)
	{
		left=beg=pop(low);
		right=en=pop(up);
		loc=beg;
		j=1;
		k=1;
		while(j)
		{
			switch(k)
			{
			case 1:
				while(a[loc]<=a[right]&&loc!=right)
```

```
                {
                        right--;
                }
                if(loc==right)
                {
                        j=0;
                        break;
                }
                if(a[right]<a[loc])
                {
                        swap1(a[loc],a[right]);
                        loc=right;
                        k=2;
                }
                break;
        case 2:
                while(a[loc]>=a[left]&&loc!=left)
                {
                        left++;
                }
                if(loc==left)
                {
                        j=0;
                        break;
                }
                if(a[left]>a[loc])
                {
                        swap1(a[loc],a[left]);
                        loc=left;
                        k=1;
                }
                break;
        }
    }
    if(beg<loc-1)
    {
        push(beg,low);
        push(loc-1,up);
    }
    if(loc+1<en)
    {
        push(loc+1,low);
        push(en,up);
    }
}
j=0;
while(j!=n)
{
    cout<<a[j]<<" ";
    j++;
```

```
    }
    cout<<"\n";
}
```

"F:\Ankit\quick sort using Stack.exe"

```
enter number of elements
6
enter values
5 1 6 9 4 6
1 4 5 6 6 9

Process returned 0 (0x0)   execution time : 14.509 s
Press any key to continue.
```

8.2(a) Write a program to find GCD of numbers using recursion.

```
#include<iostream>
using namespace std;
int gcd(int a,int b)
{
	if(a>=b)
	{
		if(a%b==0)
			return b;
		else
			return gcd(b,a%b);
	}
	else
	{
		if(b%a==0)
			return a;
		else
			return gcd(a,b%a);
	}
}
int main()
{
	cout<<"enter two no.\n";
	int a,b;
	cin>>a>>b;
	cout<<gcd(a,b);
}
```

"F:\Ankit\GCD of numbers.exe"

```
enter two no.
81 60
3
Process returned 0 (0x0)   execution time : 22.255 s
Press any key to continue.
```

8.2(b) Write a program to apply quick sort using recursion.

```
#include<iostream>
using namespace std;
void quick_sort(int a[],int first, int last)
{
	int low=first,high=last,pivot,temp;
	pivot=a[(low+high)/2];
	while(low<=high)
	{
		while(a[low]<pivot)
			low++;
		while(a[high]>pivot)
			high--;
		if(low<=high)
		{
			temp=a[low];
			a[low]=a[high];
			a[high]=temp;
			low++;
			high--;
		}
	}
	if(first<high)
		quick_sort(a,first,high);
	if(low<last)
		quick_sort(a,low,last);
}
int main()
{
	int n;
	cout<<"enter no of elements :";
	cin>>n;
	cout<<"enter elements :\n";
	int a[n],i;
	for(i=0;i<n;i++)
		cin>>a[i];
	quick_sort(a,0,n-1);
	for(i=0;i<n;i++)
		cout<<a[i]<<" ";
}
```

"F:\Ankit\quick sort using recursion.exe"

```
enter no of elements :5
enter elements :
5 3 9 4 2
2 3 4 5 9
Process returned 0 (0x0)   execution time : 10.995 s
Press any key to continue.
```

8.2(c1) Write a program to calculate factorial of a number using non tail recursion.

```
#include<iostream>
using namespace std;
int factl(int n)
{
	if(n==0)
		return 1;
	else
		return n*fact(n-1);
}
int main()
{
	int n;
	cout<<"enter the no. u want to calculate factorial :\n";
	cin>>n;
	cout<<fact(n);
}
```

"F:\Ankit\factorial using non tail recursion.exe"

```
enter the no. u want to calculate factorial :
7
5040
Process returned 0 (0x0)   execution time : 4.235 s
Press any key to continue.
```

8.2 (c2) Write a program to calculate factorial using tail recursion.

```
#include<iostream>
using namespace std;
int fact(int n,int ans=1)
{
	if(n==0)
		return ans;
	else
	{
		ans=n*ans;
		return fact(n-1,ans);
	}
}
int main()
{
	int n;
	cout<<"enter the no. u want to calculate factorial :\n";
	cin>>n;
	cout<<fact(n);
}
```

"F:\Ankit\factorial using tail recursion.exe"

```
enter the no. u want to calculate factorial :
4
24
Process returned 0 (0x0)   execution time : 2.188 s
Press any key to continue.
```

8.2(d1) Write a program to print fibonnaci series using non tail recusrion.

```
#include<iostream>
using namespace std;
int fibo(int n)
{
	if(n==1)
		return 0;
	else if(n==2)
		return 1;
	else
		return fibo(n-1)+fibo(n-2);
}
void series(int n)
{
	int i;
	cout<<"series is as follows :\n";
	for(i=1;i<=n;i++)
		cout<<fibo(i)<<" ";
}
int main()
{
	cout<<"enter the no of terms u want to print :\n";
	int n;
	cin>>n;
	series(n);
}
```

"F:\Ankit\fibonnaci series using non tail recusrion.exe"

```
enter the no of terms u want to print :
5
series is as follows :
0 1 1 2 3
Process returned 0 (0x0)   execution time : 5.251 s
Press any key to continue.
```

8.2(d2) Write a program to print fibonnaci series using tail recursion.

```
#include<iostream>
using namespace std;
int fibo(int n,int f1=0,int f2=1)
{
	if(n==0)
		return f1;
	else if(n==1)
		return f2;
	else
	{
		return fibo(n-1,f2,f2+f1);
	}
}
void series(int n)
{
	int i;
	cout<<"series is as follows :\n";
	for(i=1;i<=n;i++)
		cout<<fibo(i)<<" ";
}
int main()
{
	cout<<"enter the no of terms u want to print :\n";
	int n;
	cin>>n;
	series(n);
}
```

"F:\Ankit\fibonnaci series using tail recursion.exe"

```
enter the no of terms u want to print :
9
series is as follows :
1 1 2 3 5 8 13 21 34
Process returned 0 (0x0)   execution time : 3.531 s
Press any key to continue.
```

9.1 WAP to implement basic operations on queue using array(Insertion,Deletion and display of elements).

```
#include<iostream>
#include<stdlib.h>
using namespace std;
struct arrayqueue
{
	int top,base,cap;
	int *no;
};
arrayqueue* create(int n)
{
	arrayqueue *temp=(arrayqueue*)malloc(sizeof(arrayqueue));
	temp->top=-1;
	temp->base=-1;
	temp->cap=n;
	temp->no=(int*)malloc(sizeof(int)*n);
	return temp;
}
int full(arrayqueue *head)
{
	if((head->top+1)%(head->cap)==(head->base))
		return 1;
	else
		return 0;
}
int blank(arrayqueue *head)
{
	if(head->top == -1 && head->base==-1)
		return 1;
	else
		return 0;
}
void enqueue(arrayqueue *head,int n)
{
	if(!full(head))
	{
		head->top=((head->top)+1)%(head->cap);
		head->no[head->top]=n;
		if(head->base==-1)
			head->base=head->top;
	}
}
int dequeue(arrayqueue *head)
{
	if(!blank(head))
	{
		int n=head->no[head->base];
		if(head->base==head->top)
```

```
            {
                head->base=head->top=-1;
            }
            else
                head->base =((head->base)+1)%(head->cap);
            return n;
        }
}
void reversequeue(arrayqueue *head)
{
        int temp=0;
        if(!blank(head))
        {
            temp=dequeue(head);
            if(!blank(head))
                reversequeue(head);
        }
        enqueue(head,temp);

}
void print(arrayqueue *head)
{
        while(!blank(head))
        {
            cout<<dequeue(head)<<" ";
        }
}
int main()
{
        int i,n;
        cout<<"enter the capacity :";
        cin>>n;
        arrayqueue *q=create(n);
        cout<<"enter numbers:\n";
        for(i=0;i<n;i++)
        {
            int x;
            cin>>x;
            enqueue(q,x);
        }
        cout<<"enter the no. of elements u want to delete :";
        cin>>n;
        while(n--)
            dequeue(q);
        cout<<"final list :\n";
        print(q);
}
```

"F:\Ankit\basic operations on queue using array.exe"

```
enter the capacity :5
enter numbers:
45 1 56 52 85
enter the no. of elements u want to delete :2
final list :
56 52 85
Process returned 0 (0x0)   execution time : 25.254 s
Press any key to continue.
```

9.2 WAP to implement basic operations on queue using linked list.

```
#include<iostream>
#include<stdlib.h>
using namespace std;
struct node
{
	int data;
	struct node *next;
};
void enqueue(node **head,int n)
{
	node *new1=(node*)malloc(sizeof(node)),*temp=*head;
	new1->next=NULL;
	new1->data=n;
	if(*head==NULL)
	{
		*head=new1;
	}
	else
	{
		while(temp->next!=NULL)
			temp=temp->next;
		temp->next=new1;
	}

}
int dequeue(node** head)
{
	int no;
	node *temp=*head;
	if(temp==NULL)
	{
		cout<<"stack is empty\n";
		return -1;
	}
	else
	{
		no=temp->data;
		*head=temp->next;
		delete(temp);
		return no;
	}

}
void print(node *head)
{
	while(head!=NULL)
	{
```

```
        cout<<head->data<<"\n";
        head=head->next;
    }
}
int main()
{
    node *head=(node*)malloc(sizeof(node));
    head->next=NULL;
    int n;
    cout<<"enter the data of head\n";
    cin>>head->data;

    while(1)
    {
        cout<<"enter ur choice 1 enqueue \n 2 dequeue\n3 print \n4 exit\n";
        int ch;
        cin>>ch;
        switch(ch)
        {
        case 1:
            cout<<"enter data\n";
            cin>>n;
            enqueue(&head,n);
            break;
        case 2:
            n=dequeue(&head);
            cout<<n;
            break;
        case 3:
            print(head);
            break;
        case 4:
            exit(0);
            break;
        }
    }
}
```

"F:\Ankit\basic operations on queue using linked list.exe"

```
enter the data of head
25
enter ur choice 1 enqueue
 2 dequeue
3 print
4 exit
1
enter data
15
enter ur choice 1 enqueue
 2 dequeue
3 print
4 exit
1
enter data
36
enter ur choice 1 enqueue
 2 dequeue
3 print
4 exit
1
enter data
89
enter ur choice 1 enqueue
 2 dequeue
3 print
4 exit
2
25enter ur choice 1 enqueue
 2 dequeue
3 print
4 exit
3
15
36
89
enter ur choice 1 enqueue
 2 dequeue
3 print
4 exit
4

Process returned 0 (0x0)   execution time : 65.617 s
Press any key to continue.
```

9.3(a) WAP to implement Input restricted Queue.

```
#include<bits/stdc++.h>
using namespace std;
class queue1
{
	int cap,top,base,*no;
public:
	void create(int n)
	{
		cap=n;
		top=-1;
		base=-1;
		no=new int[n];
	}
	bool full()
	{
		if(((top+1)%cap)==base)
			return 1;
		else
			return 0;
	}
	bool blank()
	{
		if((top==-1)&&base ==-1)
			return 1;
		else
			return 0;
	}
	void enqueue(int n)
	{
		if(!full())
		{
			top=(top+1)%cap;
			no[top]=n;
			if(base==-1)
				base=top;
		}
	}
	int dequeue(int k=0)
	{
		int n;
		if(k==0)
		{
			if(!blank())
			{
				n=no[base];
				if(base==top)
					base=top=-1;
```

```
                else
                        base=(base+1)%cap;
                        return n;
                }
        }
        else if(k==1)
        {
                if(!blank())
                {
                        n=no[top];
                        if(base==top)
                                base=top=-1;
                        else
                                top=(top+cap-1)%cap;
                                return n;
                }

        }
        return -1;
    }
    void print()
    {
        while(!blank())
        {
                cout<<dequeue()<<" ";
        }
    }
};
int main()
{
    queue1 q;
    cout<<"enter the capacity of queue :";
    int n;
    cin>>n;
    q.create(n);
    cout<<"enter your choice :\n1 enqueue\n2 dequeue\n3 exit";
    while(1)
    {
        int ch,x;
        cin>>ch;
        switch(ch)
        {
        case 1:
                cout<<"enter data :";
                cin>>x;
                q.enqueue(x);
                break;
        case 2:
                cout<<"press 0 to dequeue from base and 1 from top: ";
                int k;
```

```
            cin>>k;
            cout<<q.dequeue(k);
            break;
        case 3:
            exit(0);
        }
    }

}
```

"F:\Ankit\Input restricted Queue.exe"

```
enter the capacity of queue :6
enter your choice :
1 enqueue
2 dequeue
3 exit1
enter data :25
1
enter data :36
1
enter data :56
1
enter data :78
1
enter data :10
2
press 0 to dequeue from base and 1 from top: 1
10
2
press 0 to dequeue from base and 1 from top: 0
25
2
press 0 to dequeue from base and 1 from top: 0
36
3

Process returned 0 (0x0)   execution time : 86.624 s
Press any key to continue.
```

9.3(b) Write a program to implement output restricted queue.

```
#include<bits/stdc++.h>
using namespace std;
class queue1
{
	int cap,top,base,*no;
public:
	void create(int n)
	{
		cap=n;
		top=-1;
		base=-1;
		no=new int[n];
	}
	bool full()
	{
		if(((top+1)%cap)==base)
			return 1;
		else
			return 0;
	}
	bool blank()
	{
		if((top==-1)&&base ==-1)
			return 1;
		else
			return 0;
	}
	void enqueue(int k,int n)
	{
		if(k==0)
		{
			if(!full())
			{
				top=(top+1)%cap;
				no[top]=n;
				if(base==-1)
					base=top;
			}
		}
		else if(k==1)
		{
			if(!full())
			{
				base=(base+cap-1)%cap;
				no[base]=n;
				if(top==-1)
					top=base;
```

```
                }
            }
        }
        int dequeue()
        {
            int n;
            if(!blank())
            {
                n=no[base];
                if(base==top)
                    base=top=-1;
                else
                    base=(base+1)%cap;
                    return n;
            }
            return -1;
        }
        void print()
        {
            while(!blank())
            {
                cout<<dequeue()<<" ";
            }
        }
};
int main()
{
        queue1 q;
        cout<<"enter the capacity of queue :";
        int n;
        cin>>n;
        q.create(n);
        cout<<"enter your choice :\n1 enqueue\n2 dequeue\n3 exit";
        while(1)
        {
            int ch,x;
            cin>>ch;
            switch(ch)
            {
            case 1:
                cout<<"enter data :";
                cin>>x;
                cout<<"press 0 to enqueue from top and 1 from base ";
                int k;
                cin>>k;
                q.enqueue(k,x);
                break;
            case 2:
                cout<<q.dequeue();
                break;
```

```
        case 3:
            exit(0);
        }
    }

}
```

"F:\Ankit\output restricted queue.exe"

```
enter the capacity of queue :5
enter your choice :
1 enqueue
2 dequeue
3 exit1
enter data :25
press 0 to enqueue from top and 1 from base 0
1
enter data :36
press 0 to enqueue from top and 1 from base 0
1
enter data :96
press 0 to enqueue from top and 1 from base 0
1
enter data :65
press 0 to enqueue from top and 1 from base 1
1
enter data :78
press 0 to enqueue from top and 1 from base 1
2
78
2
65
2
253

Process returned 0 (0x0)   execution time : 118.850 s
Press any key to continue.
```

9.4(a) WAP to implement Priority queue using linked list.

```
#include <iostream>
#include <stdlib.h>
using namespace std;
struct Node {
	int data;
	int priority;
	Node* next;

};
Node* newNode(int d, int p)
{
	Node* temp = (Node*)malloc(sizeof(Node));
	temp->data = d;
	temp->priority = p;
	temp->next = NULL;
	return temp;
}
int peek(Node** head)
{
	return (*head)->data;
}
void pop(Node** head)
{
	Node* temp = *head;
	(*head) = (*head)->next;
	free(temp);
}
void push(Node** head, int d, int p)
{
	Node* start = (*head);
	Node* temp = newNode(d, p);
	if ((*head)->priority > p) {
		temp->next = *head;
		(*head) = temp;
	}
	else {
		while (start->next != NULL &&
			start->next->priority < p) {
			start = start->next;
		}
		temp->next = start->next;
		start->next = temp;
	}
}
int isEmpty(Node** head)
{
	return (*head) == NULL;
}
```

```
int main()
{
	cout<<"enter no of elements:";
	int n;
	cin>>n;
	cout<<"enter data and priority :";
	int x,y,i;
	cin>>x>>y;
	Node* pq = newNode(x,y);
	for(i=0;i<n-1;i++)
	{
		cin>>x>>y;
		push(&pq,x,y);
	}
	while (!isEmpty(&pq)) {
		cout<<peek(&pq)<<" ";
		pop(&pq);
	}
	return 0;
}
#include <iostream>
#include <stdlib.h>
using namespace std;
struct Node {
	int data;
	int priority;
	Node* next;

};
Node* newNode(int d, int p)
{
	Node* temp = (Node*)malloc(sizeof(Node));
	temp->data = d;
	temp->priority = p;
	temp->next = NULL;
	return temp;
}
int peek(Node** head)
{
	return (*head)->data;
}
void pop(Node** head)
{
	Node* temp = *head;
	(*head) = (*head)->next;
	free(temp);
}
void push(Node** head, int d, int p)
{
	Node* start = (*head);
```

```
        Node* temp = newNode(d, p);
        if ((*head)->priority > p) {
                temp->next = *head;
                (*head) = temp;
        }
        else {
                while (start->next != NULL &&
                        start->next->priority < p) {
                    start = start->next;
                }
                temp->next = start->next;
                start->next = temp;
        }
}
int isEmpty(Node** head)
{
        return (*head) == NULL;
}
int main()
{
        cout<<"enter no of elements:";
        int n;
        cin>>n;
        cout<<"enter data and priority :";
        int x,y,i;
        cin>>x>>y;
        Node* pq = newNode(x,y);
        for(i=0;i<n-1;i++)
        {
                cin>>x>>y;
                push(&pq,x,y);
        }
        while (!isEmpty(&pq)) {
                cout<<peek(&pq)<<" ";
                pop(&pq);
        }
        return 0;
}
```

"F:\Ankit\Priority queue using linked list.exe"

```
enter no of elements:5
enter data and priority :10 3
95
2
12 1
35 5
15 4
12 95 10 15 35
Process returned 0 (0x0)   execution time : 36.092 s
Press any key to continue.
```

9.4(b) Write a program to implement priority queue by array.

```
#include<bits/stdc++.h>
using namespace std;
struct arrayqueue
{
	int top,base,cap;
	int *no;
};
arrayqueue* create(int n)
{
	arrayqueue *temp=(arrayqueue*)malloc(sizeof(arrayqueue));
	temp->top=-1;
	temp->base=-1;
	temp->cap=n;
	temp->no=(int*)malloc(sizeof(int)*n);
	return temp;
}
int full(arrayqueue *head)
{
	if((head->top+1)%(head->cap)==(head->base))
		return 1;
	else
		return 0;
}
int blank(arrayqueue *head)
{
	if(head->top == -1 && head->base==-1)
		return 1;
	else
		return 0;
}
void enqueue(arrayqueue *head,int n)
{
	if(!full(head))
	{
		head->top=((head->top)+1)%(head->cap);
		head->no[head->top]=n;
		if(head->base==-1)
			head->base=head->top;
	}
}
int dequeue(arrayqueue *head)
{
	if(!blank(head))
	{
		int n=head->no[head->base];
		if(head->base==head->top)
		{
			head->base=head->top=-1;
```

```
            }
            else
                head->base =((head->base)+1)%(head->cap);
            return n;
        }
        else return -1;
}
int main()
{
        int n,i,j=1;
        cout<<"enter the no. of priorities :\n";
        cin>>n;
        arrayqueue *q[n];
        i=0;
        while(i<n)
        {
            q[i]=create(10);
           i++;
        }
        while(1)
        {
            cout<<"1 to insert\n 2 to delete\n3 to display";
            cout<<"\nenter your choice\n";
            cin>>j;
            switch(j)
            {
            case 1:

                cout<<"enter the priority of element u want to    :\n";
                int x;
                cin>>x;
                cout<<"enter the element :\n";
                int y;
                cin>>y;
                enqueue(q[x-1],y);
                break;
            case 2:
                y=0;
                while(1 && y<n)
                {
                    if(blank(q[y]))
                    y++;
                    if(dequeue(q[y]))
                        break;
                }
                break;
            case 3:
                  y=0;
                while(1 && y<n)
                {
```

```
                if(blank(q[y]))
                y++;
                else
                cout<<dequeue(q[y])<<" ";
            }
            break;
        }
    }

}
```

"F:\Ankit\priority queue by array.exe"

```
3
enter the element :
25

enter your choice
1
enter the priority of element u want to  :
5
enter the element :
36

enter your choice
1
enter the priority of element u want to  :
3
enter the element :
10

enter your choice
1
enter the priority of element u want to  :
1
enter the element :
85

enter your choice
3
85 25 10 36
```

10.1 WAP to implement the basic operations on Binary tree

a. Insertion
b. Deletion
c. Counting Number of nodes
d. Height of tree

```
#include<iostream>
using namespace std;
struct node
{
	int info;
	node *left,*right;
};
 node* create_node(int a)
	{
		node *n;
		n=new node;
		n->info=a;
		n->left='\0';
		n->right='\0';
		return n;
	}

	void set_node(int a,node *p,node *r,char c)
	{
		int k=0;
		if(a==r->info)
		{
			k=1;
			if(c=='l')
				r->left=p;
			else
				r->right=p;
		}
		if(k==0)
		{
			if(r->left!='\0')
				set_node(a,p,r->left,c);
			if(r->right!='\0')
				set_node(a,p,r->right,c);
		}
	}
	void display(node *r)
	{
		cout<<r->info<<" ";
		if(r->left!='\0')
			display(r->left);
		if(r->right!='\0')
```

```
            display(r->right);
}
node *x='\0';
void find_node(int a,node *r)
{
      if(r->left->info==a||r->right->info==a)
            x=r;
      else
      {
            if(r->left!='\0')
                  find_node(a,r->left);
            if(r->right!='\0')
                  find_node(a,r->right);
      }
}
void delete_node(int a,node *r)
{
      node *n,*p;
      int k=1;
      find_node(a,r);
      cout<<"hello\n";
      n=x;
      p=n->left;
      if(p->info!=a)
      {
            p=n->right;
            k=2;
      }
      cout<<n->info<<" "<<p->info<<" ";
     if(p->left=='\0'&&p->right=='\0')
     {
           delete p;
           if(k==1)
             n->left='\0';
           else
             n->right='\0';
     }
     else if(p->left=='\0')
     {
           if(k==1)
             n->left=p->right;
           else
             n->right=p->right;
             delete p;
     }
     else if(p->right=='\0')
     {
           if(k==1)
             n->left=p->left;
           else
```

```
                n->right=p->left;
                delete p;
            }
        }
        int maximum(int a,int b)
        {
            if(a>=b)
                return a;
            else
                return b;
        }
        int height(node *r)
        {
            if(r=='\0')
                return 0;
            else
                return (1+maximum(height(r->left),height(r->right)));
        }
int main()
{
        node *root,*p;
        int n,a,b,i,j,k;
        char c;
        cout<<"press 1 for create a tree\npress 2 for insert a node\npress 3 for preorder display\n";
        cout<<"press 4 for delete a node\npress 5 for find height of tree\npress 6 for exit\n";
        cout<<"enter choice\n";
        cin>>j;
        while(j!=6)
        {
            switch(j)
            {
            case 1:
                cout<<"enter total number of nodes\n";
                cin>>n;
                cout<<"enter root node\n";
                cin>>a;
                root=create_node(a);
                k=n;
                k=k-1;
                  while(k--)
                  {
                        cout<<"enter parent node,node info and side(l or r)\n";
                        cin>>a>>b>>c;
                        p=create_node(b);
                        set_node(a,p,root,c);
                   }
                   break;
            case 2:
                cout<<"enter parent node,node info and side(l or r)\n";
                cin>>a>>b>>c;
```

```
            p=create_node(b);
            set_node(a,p,root,c);
            break;
        case 3:
            display(root);
            cout<<"\n";
            break;
        case 4:
            cout<<"enter node info\n";
            cin>>a;
            delete_node(a,root);
            break;
        case 5:
            a=height(root);
            cout<<"height of the tree is "<<a<<"\n";
            break;
        }
        cout<<"enter choice\n";
        cin>>j;
    }
}
```

"F:\DSA 2nd\binary tree.exe"

```
press 1 for create a tree
press 2 for insert a node
press 3 for preorder display
press 4 for delete a node
press 5 for find height of tree
press 6 for exit
enter choice
1
enter total number of nodes
5
enter root node
5
enter parent node,node info and side(l or r)
5 4 l
enter parent node,node info and side(l or r)
5 8 r
enter parent node,node info and side(l or r)
4 3 l
enter parent node,node info and side(l or r)
4 9 r
enter choice
3
5 4 3 9 8
enter choice
2
enter parent node,node info and side(l or r)
8 7 l
enter choice
3
5 4 3 9 8 7
enter choice
5
height of the tree is 3
enter choice
6

Process returned 0 (0x0)   execution time : 116.558 s
Press any key to continue.
```

10.2 WAP to implement the basic operations on Binary Search tree

a. Insertion
b. Deletion
c. Counting Number of Nodes
d. Height of tree

```
#include<iostream>
using namespace std;
struct node
{
	int num;
	node*l,*r;
};
node*f(int);
void inorder(node*);
int height(node*);
node* insert1(node*,int);
node* delete1(node*,int);
int main()
{
	node*root=0;
	int e;
	root=f(4);
	root->l=f(2);
	root->r=f(5);
	root->l->l=f(1);
	root->l->r=f(3);
	cout<<"inorder ";
	inorder(root);
	cout<<"\nheight="<<height(root)<<"\n";
	cout<<"enter item to insert";
	cin>>e;
	root=insert1(root,e);
	inorder(root);
	cout<<"enter the item to delete";
	cin>>e;
	root=delete1(root,e);
	inorder(root);
	return 0;
}
node*f(int a)
{
	node*n=new node;
	n->num=a;
	n->l=0;
	n->r=0;
	return n;
}
```

```
void inorder(node*p)
{
	if(p==0)
		return;
	inorder(p->l);
	cout<<p->num<<" ";
	inorder(p->r);
}
int height(node*p)
{
	if(p==0)
		return 0;
	return 1+max(height(p->l),height(p->r));
}
node* insert1(node*p,int a)
{
	if(p==0)
		return f(a);
	else if(a>p->num)
		p->r=insert1(p->r,a);
	else
		p->l=insert1(p->l,a);
}
node* delete1(node*p,int a)
{
	if(p==0)
		return p;
	else if(a>p->num)
		p->r=delete1(p->r,a);
	else if(a<p->num)
		p->l=delete1(p->l,a);
	else
	{
		if(p->l!=0&&p->r!=0)
		{
			node*c=p->r;
			while(c->l!=0)
				c=c->l;
			p->num=c->num;
			p->r=delete1(p->r,c->num);
		}
		else if(p->l!=0)
			p=p->l;
		else
			p=p->r;
	}
   return p;}
```

"F:\DSA 2nd\f.exe"

```
inorder 1 2 3 4 5
height=3
enter item to insert
9
1 2 3 4 5 9
enter the item to delete
3
1 2 4 5 9
Process returned 0 (0x0)   execution time : 8.568 s
Press any key to continue.
```

11. WAP to implement:

A. Insertion in AVL tree

B. In-order Traversal of AVL tree

C. Balance Factor of AVL Tree

```
#include<bits/stdc++.h>
using namespace std;
struct avlnode
{
	int data;
	avlnode *left,*right;
};
int height(avlnode *head)
{
	if(head==NULL)
		return 0;
	else
		return(1+max(height(head->left),height(head->right)));
}
int balance_factor(avlnode *head)
{
	return (height(head->left)-height(head->right));
}
avlnode* llRotation(avlnode *head)
{
	avlnode *temp=head->left;
	head->left=temp->right;
	temp->right=head;
	return temp;
}
avlnode* rrRotation(avlnode *head)
{
	avlnode *temp=head->right;
	head->right=temp->left;
	temp->left=head;
	return temp;
}
avlnode* lrRotation(avlnode *head)
{
	head->left=rrRotation(head->left);
	return llRotation(head);
}
avlnode* rlRotation(avlnode *head)
{
	head->right=llRotation(head->right);
	return rrRotation(head);
}
avlnode* insertion(avlnode *head,int n)
{
	if(!head)
```

```
        {
            head=new avlnode;
            head->data=n;
            head->left=NULL;
            head->right=NULL;
        }
        else if(n<head->data)
        {
            head->left=insertion(head->left,n);
            if(height(head->left)-height(head->right)==2)
                if(n<head->left->data)
                    head=llRotation(head);
                else
                    head=lrRotation(head);
        }
        else if(n>head->data)
        {
            head->right=insertion(head->right,n);
            if(height(head->left)-height(head->right)==-2)
                if(n>head->right->data)
                    head=rrRotation(head);
                else
                    head=rlRotation(head);
        }
        return head;
    }
    void inorder(avlnode *head)
    {
        if(head)
        {
            inorder(head->left);
            cout<<head->data<<" ";
            inorder(head->right);
        }
    }
    int main()
    {
        int n;
        cout<<"enter the no of elements :";
        cin>>n;
        avlnode *head=NULL;
        cout<<"enter data :";
        while(n--)
        {
            int x;
            cin>>x;
            head=insertion(head,x);
        }
        cout<<"inorder of avl tree is: \n";
        inorder(head);
```

}

"C:\Users\Ankit Goyal\Downloads\avl insertion book.exe"

```
enter the no of elements :6
enter data :45 26 25 5 25 4
inorder of avl tree is:
4 5 25 26 45
Process returned 0 (0x0)   execution time : 11.227 s
Press any key to continue.
```

12.1 WAP to implement operations:

A. **Insertion in Heap Tree**
B. **Deletion in Heap tree**
C. **Deletion of Root Node from Heap**
D. **Sort the heap**

```
#include<bits/stdc++.h>
using namespace std;
void heapisation(int *no,int n,int i)
{
	if(i>=1 && i<n)
	{
		if(no[i]>no[(i-1)/2])
		{
			int temp=no[i];
			no[i]=no[(i-1)/2];
			no[(i-1)/2]=temp;
			i=(i-1)/2;
			heapisation(no,n,i);
		}
	}
}
int deletion(int *no,int *n,int i)
{
	int x=no[i],temp;
	no[i]=no[*n];
	if(2*i+2<*n)
	{
		if(no[i]<no[2*i+1] || no[i]<no[2*i+2])
		{
			temp=no[i];
			no[i]=max(no[2*i+1],no[2*i+2]);
			if(no[i]==no[2*i+1])
			{
				no[2*i+1]=temp;
				i=2*i+1;
			}
			else
			{
				no[2*i+2]=temp;
				i=2*i+2;
			}
			deletion(no,n,i);
		}
	}
	else if(2*i+1<*n)
	{
		if(no[i]<no[2*i+1])
		{
```

```
                temp=no[i];
                no[i]=(no[2*i+1]);
                no[2*i+1]=temp;
                i=2*i+1;
                deletion(no,n,i);
            }
        }
        return x;
}
void print(int *no,int n)
{
        int i;
        for(i=0;i<n;i++)
            cout<<no[i]<<" ";
        cout<<"\n";
}
int main()
{
        int no[100],n,i,x;
        cout<<"enter no. of elements :";
        cin>>n;
        x=n;
        for(i=0;i<n;i++)
        {
            cin>>no[i];
            heapisation(no,n,i);
        }
        cout<<"sorted array is :\n";
        for(i=0;i<x;i++)
        {
            n=n-1;
            cout<<deletion(no,&n,0)<<" ";
        }
}
```

"F:\Ankit\heap sort.exe"

```
enter no. of elements :5
4 8 9 6 2
sorted array is :
9 8 6 4 2
Process returned 0 (0x0)   execution time : 6.014 s
Press any key to continue.
```

12.2 WAP to implement:

A. **Creation of Graph**
B. **Insertion of an edge**
C. **Deletion of an edge**
D. **Display of graph**

```
#include<bits/stdc++.h>
using namespace std;
struct graph
{
	int v,e;
	int **adj;
};
graph* create_graph()
{
	int i,j,x,y;
	graph *G;
	G=new graph;
	cout<<"enter no. of nodes and edges: ";
	cin>>G->v>>G->e;
	G->adj=(int**)malloc(sizeof(int*)*G->v);
	for(i=0;i<G->v;i++)
		G->adj[i]=(int*)malloc(sizeof(int)*G->v);
	for(i=0;i<G->v;i++)
		for(j=0;j<G->v;j++)
			G->adj[i][j]=0;
	cout<<"enter starting node and ending node for all edge: ";
	for(i=0;i<G->e;i++)
	{
		cin>>x>>y;
		G->adj[x][y]=1;
		G->adj[y][x]=1;
	}
	return G;
}
main()
{
	int ch;
	graph *G;
	G=create_graph();

	while(1)
	{
		cout<<"1 for insertion of edge\n 2 for deletion of edge\n 3 to display matrix\n enter your
choice: ";
		cin>>ch;
		switch(ch)
		{
```

```
        case 1:
            int x,y;
            cout<<"enter starting node and ending node for all edge: ";
            cin>>x>>y;
            if(G->adj[x][y]==0)
                {G->adj[x][y]=1;G->adj[y][x]=1;}
            else
                cout<<"already present\n";
            break;
        case 2:
            cout<<"enter starting node and ending node for all edge: ";
            cin>>x>>y;
            if(G->adj[x][y]==1)
                {G->adj[x][y]=0;G->adj[y][x]=0;}
            else
                cout<<"already not present\n";
            break;
        case 3:
            cout<<"adjency matrix is \n";
            for(int i=0;i<G->v;i++)
            {
                for(int j=0;j<G->v;j++)
                    cout<<G->adj[i][j]<<"     ";
                cout<<"\n";
            }
            break;
        }
    }
```

"F:\Ankit\graph implementation.exe"

```
enter no. of nodes and edges: 5 6
enter starting node and ending node for all edge: 0 1
1 2
0 4
2 4
2 3
3 4
1 for insertion of edge
 2 for deletion of edge
 3 to display matrix
 enter your choice: 2
enter starting node and ending node for all edge: 0 4
1 for insertion of edge
 2 for deletion of edge
 3 to display matrix
 enter your choice: 3
adjency matrix is
0   1   0   0   0
1   0   1   0   0
0   1   0   1   1
0   0   1   0   1
0   0   1   1   0
```

}

13.1(a) WAP to find Path matrix using Powers of matrix.

```
#include<bits/stdc++.h>
using namespace std;
struct node
{
	int data,state;
};
struct graph
{
	int v,e,**matrix;
	node **nodes;
};
graph* create()
{
	graph *temp=new graph;
	int i,j,x;
	cout<<"enter the no. of vertices :";
	cin>>temp->v;
	temp->matrix=(int**)malloc(sizeof(int)*(temp->v));
	for(i=0;i<temp->v;i++)
		temp->matrix[i]=new int[temp->v];
	int mat[temp->v][temp->v];
	temp->nodes=(node**)malloc(sizeof(node)*temp->v);
	for(i=0;i<temp->v;i++)
	{
		node *y=new node;
		cout<<"enter the data of node :";
		cin>>y->data;
		y->state=0;
		temp->nodes[i]=y;
	}
	cout<<1;
	for(i=0;i<temp->v;i++)
		for(j=0;j<temp->v;j++)
			temp->matrix[i][j]=0;
	  for(i=0;i<temp->v;i++)
		temp->matrix[i][i]=0;
	    cout<<"enter the no of edges :";
	cin>>temp->e;
	cout<<"enter source and destination of edges:";
	for(x=0;x<temp->e;x++)
	{
		cin>>i>>j;
		int k;
		   temp->matrix[i-1][j-1]=1;
		}
	return temp;
}
```

```
int** path(graph *temp)
{
	int i,j,k,l,t[temp->v][temp->v];
	int **path=(int**)malloc(sizeof(int)*(temp->v));
	for(i=0;i<temp->v;i++)
		path[i]=new int[temp->v];
	int **path1=(int**)malloc(sizeof(int)*(temp->v));
	for(i=0;i<temp->v;i++)
		path1[i]=new int[temp->v];
	for(i=0;i<temp->v;i++)
	{
		for(j=0;j<temp->v;j++)
		{
		    path[i][j]=temp->matrix[i][j];
		      path1[i][j]=temp->matrix[i][j];
		}
	}
	for(l=0;l<(temp->v)-1;l++)
	{
		for(i=0;i<temp->v;i++)
		{
			for(j=0;j<temp->v;j++)
			{
				int sum=0;
				for(k=0;k<temp->v;k++)
				{
					sum=sum+(path[i][k]*(temp->matrix[k][j]));
				}
				t[i][j]=sum;
			}
		}
		for(int m=0;m<temp->v;m++)
		{
			for(int n=0;n<temp->v;n++)
			{
				path[m][n]=t[m][n];
				path1[m][n]=path[m][n]+path1[m][n];
			}
		}

	}
	return path1;
}
int main()
{
	int i,j;
	graph *head=create();
	for(i=0;i<head->v;i++)
	{
		for(j=0;j<head->v;j++)
```

```
        {
            cout<<head->matrix[i][j]<<" ";
        }
        cout<<"\n";
    }
    int **p=path(head);
    cout<<"\npath matrix :\n";
    for(i=0;i<head->v;i++)
    {
        for(j=0;j<head->v;j++)
        {
            cout<<p[i][j]<<" ";
        }
        cout<<"\n";
    }
}
```

"F:\Ankit\path matrix.exe"

```
enter the no. of vertices :5
enter the data of node :10
enter the data of node :11
enter the data of node :14
enter the data of node :16
enter the data of node :8
1enter the no of edges :6
enter source and destination of edges:1 2
2 3
3 4
4 5
1 5
5 3
0 1 0 0 1
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
0 0 1 0 0

path matrix :
0 1 4 2 3
0 0 2 2 1
0 0 1 2 2
0 0 2 1 2
0 0 2 2 1

Process returned 0 (0x0)   execution time : 85.796 s
Press any key to continue.
```

13.1(b) WAP to find Path matrix using Warshall algorithm.

```
#include<bits/stdc++.h>
using namespace std;
struct node
{
	int data,state;
};
struct graph
{
	int v,e,**matrix;
	node **nodes;
};
graph* create()
{
	graph *temp=new graph;
	int i,j,x;
	cout<<"enter the no. of vertices :";
	cin>>temp->v;
	temp->matrix=(int**)malloc(sizeof(int)*(temp->v));
	for(i=0;i<temp->v;i++)
		temp->matrix[i]=new int[temp->v];
	int mat[temp->v][temp->v];
	temp->nodes=(node**)malloc(sizeof(node)*temp->v);
	for(i=0;i<temp->v;i++)
	{
		node *y=new node;
		cout<<"enter the data of node :";
		cin>>y->data;
		y->state=0;
		temp->nodes[i]=y;
	}
	cout<<1;
	for(i=0;i<temp->v;i++)
		for(j=0;j<temp->v;j++)
			temp->matrix[i][j]=99999;
	  for(i=0;i<temp->v;i++)
		temp->matrix[i][i]=0;
	//temp->matrix=(int**)mat;
	cout<<"enter the no of edges :";
	cin>>temp->e;
	cout<<"enter source and destination of edges:";
	for(x=0;x<temp->e;x++)
	{
		cin>>i>>j;
		int k;
		cout<<"enter the weight of node:";
		cin>>k;
		temp->matrix[i-1][j-1]=k;
		//temp->matrix[j-1][i-1]=1;
```

```
        }
        return temp;
}
void floyd_short(graph *head)
{
        int i,j,k;
        for(i=0;i<head->v;i++)
        {
                for(j=0;j<head->v;j++)
                {
                        for(k=0;k<head->v;k++)
                        {
                                if(head->matrix[i][k]+head->matrix[k][j]<head->matrix[i][j])
                                {
                                        head->matrix[i][j]=head->matrix[i][k]+head->matrix[k][j];
                                }
                        }
                }
        }
}
int main()
{
        graph *head=create();
        floyd_short(head);
        for(int i=0;i<head->v;i++)
        {
                for(int j=0;j<head->v;j++)
                {
                        cout<<setw(10)<<head->matrix[i][j]<<" ";
                }
                cout<<"\n";
        }
}
```

"F:\Ankit\floyd warshall.exe"

```
enter the no. of vertices :5
enter the data of node :10
enter the data of node :11
enter the data of node :14
enter the data of node :16
enter the data of node :8
1enter the no of edges :6
enter source and destination of edges:1 2
enter the weight of node:10
2 3
enter the weight of node:11
3 4
enter the weight of node:14
4 5
enter the weight of node:16
1 5
enter the weight of node:8
5 3
enter the weight of node:5
         0         10         13         27          8
     99999          0         11         25         41
     99999      99999          0         14         30
     99999      99999         21          0         16
     99999      99999          5         19          0

Process returned 0 (0x0)   execution time : 162.641 s
Press any key to continue.
```

13.2 Write Functions to perform:

A. Breadth First Search
B. Depth First Search

```
#include<bits/stdc++.h>
using namespace std;
struct node
{
	int data,state;
};
struct graph
{
	int v,e,**matrix;
	node **nodes;
};
struct arraystack
{
	int top,cap;
	int *no;
};
struct arrayqueue
{
	int base,rear,cap,*no;
};
arrayqueue* create_q(int n)
{
	arrayqueue *temp=new arrayqueue;
	temp->base=-1;
	temp->rear=-1;
	temp->cap=n;
	temp->no=new int[n];
}
bool full(arrayqueue *head)
{
	if((head->rear+1)%head->cap==head->base)
		return 1;
	return 0;
}
bool blank(arrayqueue *head)
{
	if(head->base==-1)
		return 1;
	else
		return 0;
}
void enqueue(arrayqueue *head,int n)
{
	if(!full(head))
```

```
        {
            head->rear=(head->rear+1)%head->cap;
            head->no[head->rear]=n;
            if(head->base==-1)
                head->base=head->rear;
        }
}
int dequeue(arrayqueue *head)
{
        if(!blank(head))
        {
            int n=head->no[head->base];
            if(head->base==head->rear)
                head->base=head->rear=-1;
            else
                head->base=(head->base+1)%head->cap;
                return n;
        }
}
arraystack* create(int n)
{
        arraystack *temp=new arraystack;
        temp->top=-1;
        temp->cap=n;
        temp->no=(int*)malloc(sizeof(int)*n);
        return temp;
}
bool full(arraystack *head)
{
        if(head->top+1==head->cap)
            return 1;
        return 0;
}
bool blank(arraystack *head)
{
        if(head->top==-1)
            return 1;
        return 0;
}
void push(arraystack *head,int n)
{
        if(!full(head))
        {
            head->top++;
            head->no[head->top]=n;
        }
}
int pop(arraystack *head)
{
        int n;
```

```
        if(!blank(head))
        {
                n=head->no[head->top];
                head->top--;
                return n;
        }
}
graph* create()
{
        graph *temp=new graph;
        int i,j,x;
        cout<<"enter the no. of vertices :";
        cin>>temp->v;
        temp->matrix=(int**)malloc(sizeof(int)*(temp->v));
        for(i=0;i<temp->v;i++)
                temp->matrix[i]=new int[temp->v];
        int mat[temp->v][temp->v];
        temp->nodes=(node**)malloc(sizeof(node)*temp->v);
        for(i=0;i<temp->v;i++)
        {
                node *y=new node;
                cout<<"enter the data of node :";
                cin>>y->data;
                y->state=0;
                temp->nodes[i]=y;
        }
        for(i=0;i<temp->v;i++)
                for(j=0;j<temp->v;j++)
                        temp->matrix[i][j]=0;
        //temp->matrix=(int**)mat;
        cout<<"enter the no of edges :";
        cin>>temp->e;
        cout<<"enter source and destination of edges:";
        for(x=0;x<temp->e;x++)
        {
                cin>>i>>j;
                temp->matrix[i-1][j-1]=1;
                temp->matrix[j-1][i-1]=1;
        }
        return temp;
}
void dfs(graph *head)
{
        int i;
        arraystack *s=create(head->v);
        for(i=0;i<head->v;i++)
        {
                if(head->nodes[i]->state==0)
                {
                        head->nodes[i]->state=1;
```

```
                push(s,i);
            }
            while(!blank(s))
            {
                int x=pop(s),j;
                cout<<head->nodes[x]->data<<" ";
                head->nodes[x]->state=2;
                for(j=0;j<head->v;j++)
                {
                    //cout<<head->matrix[x][j];
                    if(head->matrix[x][j]==1)
                    {
                        //cout<<2;
                        if(head->nodes[j]->state==0)
                        {
                        head->nodes[j]->state=1;
                        push(s,j);
                        }
                    }
                }
            }
        }
}
void bfs(graph *head)
{
    int i,j;
    arrayqueue *q=create_q(head->v);
    for(i=0;i<head->v;i++)
    {
        if(head->nodes[i]->state==0)
        {
            head->nodes[i]->state=1;
            enqueue(q,i);
        }
        while(!blank(q))
        {
            int x=dequeue(q);
            if(head->nodes[x]->state!=2)
            {
                cout<<head->nodes[x]->data<<" ";
                head->nodes[x]->state=2;
            }
            for(j=0;j<head->v;j++)
            {
                if(head->matrix[x][j]==1)
                {
                    if(head->nodes[j]->state==0)
                    {
                        head->nodes[i]->state=1;
                        enqueue(q,j);
```

```
                        }
                    }
                }
            }
        }
}
int main()
{
    graph *head=create();
    cout<<"bfs:\n";
    bfs(head);
    int i=0;
    while(i<head->v)
    {
        head->nodes[i]->state=0;
        i++;
    }
    cout<<"\n";
    cout<<"dfs:\n";
    dfs(head);
}
```

"F:\Ankit\graph by matrix.exe"

```
enter the no. of vertices :5
enter the data of node :10
enter the data of node :11
enter the data of node :14
enter the data of node :16
enter the data of node :8
enter the no of edges :6
enter source and destination of edges:1 2
2 3
3 4
4 5
5 1
3 5
bfs:
10 11 8 14 16
dfs:
10 8 16 14 11
Process returned 0 (0x0)   execution time : 26.539 s
Press any key to continue.
```

13.3 Find the shortest path in graph using Dijkstra's algorithm.

```
#include<bits/stdc++.h>
using namespace std;
struct node
{
	int data,state;
};
struct graph
{
	int v,e,**matrix;
	node **nodes;
};
graph* create()
{
	graph *temp=new graph;
	int i,j,x;
	cout<<"enter the no. of vertices :";
	cin>>temp->v;
	temp->matrix=(int**)malloc(sizeof(int)*(temp->v));
	for(i=0;i<temp->v;i++)
		temp->matrix[i]=new int[temp->v];
	int mat[temp->v][temp->v];
	temp->nodes=(node**)malloc(sizeof(node)*temp->v);
	for(i=0;i<temp->v;i++)
	{
		node *y=new node;
		cout<<"enter the data of node :";
		cin>>y->data;
		y->state=0;
		temp->nodes[i]=y;
	}
	for(i=0;i<temp->v;i++)
		for(j=0;j<temp->v;j++)
			temp->matrix[i][j]=0;
	cout<<"enter the no of edges :";
	cin>>temp->e;
	cout<<"enter source and destination of edges:";
	for(x=0;x<temp->e;x++)
	{
		cin>>i>>j;
		cout<<"enter weight :";
		int k;
		cin>>k;
		temp->matrix[i-1][j-1]=k;
	}
	return temp;
}
int minDistance(int dist[], bool sptSet[],int n)
{
```

```
    int min = INT_MAX, min_index;

    for (int v = 0; v < n; v++)
       if (sptSet[v] == false && dist[v] <= min)
             min = dist[v], min_index = v;

    return min_index;
}
int print(int dist[], int n)
{
    cout<<("Vertex      Distance from Source\n");
    for (int i = 0; i < n; i++)
         cout<<i<<" distance ="<<dist[i]<<" \n";
}
void dijkstra(graph* head, int src)
{
      int V=head->v;
       int dist[V];
       bool sptSet[V];
       for (int i = 0; i < V; i++)
            dist[i] = INT_MAX, sptSet[i] = false;
       dist[src] = 0;
       for (int count = 0; count < V-1; count++)
       {
          int u = minDistance(dist, sptSet,head->v);
          sptSet[u] = true;
          for (int v = 0; v < V; v++)
             if (!sptSet[v] && head->matrix[u][v] && dist[u] != INT_MAX
                                                  && dist[u]+head->matrix[u][v] < dist[v])
                   dist[v] = dist[u] + head->matrix[u][v];
       }
       print(dist, V);
}
int main()
{

      graph *head=create();
      dijkstra(head, 0);
      return 0;
}
```

F:\Ankit\Dijkstra.exe

```
enter the no. of vertices :5
enter the data of node :10
enter the data of node :11
enter the data of node :14
enter the data of node :16
enter the data of node :8
enter the no of edges :6
enter source and destination of edges:1 2
enter weight :10
2 3
enter weight :11
3 4
enter weight :14
4 5
enter weight :16
1 5
enter weight :8
5 3
enter weight :5
Vertex   Distance from Source
0 distance =0
1 distance =10
2 distance =13
3 distance =27
4 distance =8

Process returned 0 (0x0)   execution time : 71.528 s
Press any key to continue.
```