Name-Neetu Kumawat

Branch-CSE-DD

Roll No.-185545

Semester-4

Subject-Lab Assignment

**Ques 1: ) Write a program to sort an array (make a dynamic array) using Bubble sort. Use 1-bit variable FLAG to signal when no interchange take place during pass. If FLAG is 0 after any pass, then list is already sorted and there is no need to continue.**

#include<iostream>

using namespace std;

void bubble\_sort(int arr[],int n)

{

int i, j, temp;

for(i = 0; i < n; i++)

{

for(j = 0; j < n-i-1; j++)

{

int flag = 0;

if( arr[j] > arr[j+1])

{

temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

flag = 1;

}

}

int flag;

if(!flag)

{

break;

}

}

cout << "Sorted Array: " ;

for(i = 0; i < n; i++)

{

cout << arr[i] << " ";

}

}

int main()

{

int n,\*a=NULL;

cout<<"Enter the size of array"<<endl;

cin>> n;

a=new int[n];

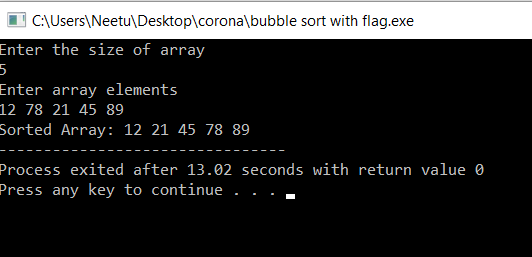
cout << "Enter array elements"<<endl;

for(int i=0;i<n;i++)

cin >> a[i];

bubble\_sort(a,n);

}



**Ques 2:) WAP to search an ITEM (integer) in an array using binary search, if FOUND then delete that item from array and if NOT FOUND than insert that item in kth position (Input k from user).**

#include<iostream>

using namespace std;

int binary(int l,int h,int item,int n,int a[])

{

int mid=(l+h)/2;

while(l<=h)

{

if(item==a[mid])

{

return mid;

}

else if(item>a[mid])

return binary(mid+1,n-1,item);

else

return binary(0,mid-1,item);

}

}

int main()

{

int item,\*a=NULL,n,loc=-1,k,data;

cout<<"enter the number of elements in an array\n";

cin>>n;

a=new int[n+1];

cout<<"enter elements of the array\n";

for(int i=0;i<n;i++)

cin>>a[i];

cout<<"enter the element which you want to search\n";

cin>>item;

loc=binary(0,n-1,item);

if(loc!=-1)

{

cout<<"item is found at location "<<loc<<endl;

cout<<"deleted element is "<<a[loc]<<" at location "<<loc<<endl;

}

else

{

cout<<"item is not present\n";

cout<<"enter the position at which you want to insert the element "<<endl;

cin>>k;

cout<<"enter the element which you want to insert in the array at given position "<<endl;

cin>>data;

for(int i=n;i<=n-k;i++)

a[i+1]=a[i];

a[k-1]=a[k];

cout<<"new array is:"<<endl;

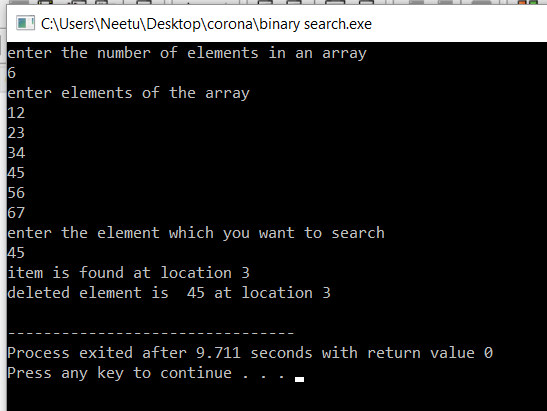
for(int i=0;i<n+1;i++)

cout<<a[i];

}

return 0;

}



**Ques 3:) WAP to enter records of Five students, which should contain \_elds like roll No., name,CGPI, semester.**

**(a) List all record of all students having CGPI greater than k.**

**(b) Insert a new record of student at kth position and print the final record.**

#include<iostream>

#include<string>

using namespace std;

struct student

{

string name;

int rollno;

float cgpi;

int semester;

}s[6];

void enterdetails()

{

cout<<"\*\*\*\*\*\*\*\* ENTER DETAILS OF THE STUDENTS\*\*\*\*\*\*\*\*"<<endl;

for(int i=0;i<5;i++)

{

cout<<"enter the name of student "<<i+1<<" ";

cin>>s[i].name;

cout<<"enter the ROLL Number of student "<<i+1<<" ";

cin>>s[i].rollno;

cout<<"enter the semester of student "<<i+1<<" ";

cin>>s[i].semester;

cout<<"enter the CGPI of student "<<i+1<<" ";

cin>>s[i].cgpi;

}

}

void showdetails()

{ int k;

cout<<"\*\*\*\*\*\*\*DETAILS OF THE STUDENTS\*\*\*\*\*\*\*\*"<<endl;

for(int i=0;i<5;i++)

{

cout<<"Name of the student "<<i+1<<" is: "<<s[i].name<<endl;

cout<<"Roll Number of the student "<<i+1<<" is: "<<s[i].rollno<<endl;

cout<<"Semester of the student "<<i+1<<" is: "<<s[i].semester<<endl;

cout<<"CGPI of the student "<<i+1<<" is: "<<s[i].cgpi<<endl;

}

}

void part1()

{

int k;

cout<<"\*\*\*\*\*\*STUDENT WITH CGPI GREATER THAN K\*\*\*\*\*\*\*"<<endl;

cout<<"enter the value of k(CGPI greater than k):\n";

cin>>k;

cout<<"list of student CGPI greater than "<<k<<endl;

for(int i=0;i<5;i++)

{

if(s[i].cgpi>k)

{

cout<<"student Nmae: "<<s[i].name<<endl;

cout<<"student ROLL Nummber: "<<s[i].rollno<<endl;

cout<<"student semester: "<<s[i].semester<<endl;

cout<<"student CGPI: "<<s[i].cgpi<<endl;

}

}

}

void part2()

{

cout<<"\*\*\*\*\*\*INSERT NEW RECORD AT pth POSITION\*\*\*\*\*\*\*\*"<<endl;

int p;

struct student st;

for(int i=4;i>=p-1;i--)

s[i+1]=s[i];

cout<<"enter the Pth position"<<endl;

cin>>p;

cout<<"enter the details: \n";

cout<<"enter the name of student "<<" ";

cin>>st.name;

cout<<"enter the ROLL Number of student "<<" ";

cin>>st.rollno;

cout<<"enter the semester of student "<<" ";

cin>>st.semester;

cout<<"enter the CGPI of student "<<" ";

cin>>st.cgpi;

cout<<"after insert a new record "<<endl;

s[i+1]=st;

for(int i=0;i<6;i++)

{

cout<<"student Nmae: "<<s[i].name<<endl;

cout<<"student ROLL Nummber: "<<s[i].rollno<<endl;

cout<<"student semester: "<<s[i].semester<<endl;

cout<<"student CGPI: "<<s[i].cgpi<<endl;

}

}

int main()

{

enterdetails();

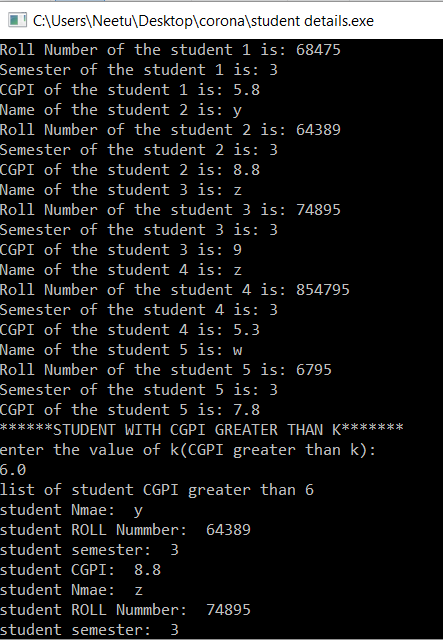
showdetails();

part1();

part2();

return 0;

}



**Ques 4:) Implement linked list and insert and delete an element into the list.**

#include<iostream>

#include<stdlib.h>

using namespace std;

struct node

{

int data;

struct node \*next;

};

typedef struct node \*lptr ;

lptr insertatending(lptr L , int data)

{

lptr temp,head;

head=L;

temp=new struct node();

temp->data=data;

temp->next=NULL;

if(L==NULL)

return temp;

while(L->next!=NULL)

L=L->next;

L->next=temp;

return head;

}

lptr insertatbeggining(lptr L,int data)

{

lptr temp=new node();

temp->data=data;

temp->next=L;

L=temp;

return L;

}

lptr insertatparticular(lptr L,int data,int knowndata)

{

lptr temp=new struct node();

lptr head=L;

temp->data=data;

while(L->data!=knowndata)

{

L=L->next;

}

temp->next=L->next;

L->next=temp;

return head;

}

lptr deletenodespecificposition(lptr L,int x)

{

lptr head=L;

if(L->next==NULL)

{

L=NULL;

}

lptr temp=L;

if(temp->data==x)

{

temp=temp->next;

L=temp;

return L;

}

while(temp)

{

if(temp->data==x)

{

temp->data=temp->next->data;

temp->next=temp->next->next;

break;

}

temp=temp->next;

}

return head;

}

lptr deletefrombeggining(lptr L)

{

lptr temp;

temp=L;

temp=temp->next;

return temp;

}

lptr deletefromending(lptr L)

{

lptr temp=new struct node();

lptr head;

if(L==NULL)

return NULL;

temp=L;

while(temp->next->next!=NULL)

{

temp=temp->next;

}

delete(temp->next);

temp->next=NULL;

return L;

}

void printlist( lptr L )

{ cout<<"link list elements are: "<<endl;

while(L != NULL)

{

cout<< L->data <<endl;

L=L->next ;

}

}

int main()

{

lptr L = NULL;

lptr p;

cout<<"Enter number of elements : "<<endl;

int x,n,data,knowndata,d;

cin>>n;

int a[50];

cout<<"Enter the elements of linked list : "<<endl;

for(int i=0 ; i<n ; i++)

{

cin>>x;

L= insertatending( L ,x) ;

}

printlist(L);

cout<<"enter data which you want to insert"<<endl;

cin>>data;

cout<<"after which node you want to insert a new node:"<<endl;

cin>>knowndata;

L=insertatparticular(L,data,knowndata);

printlist(L);

cout<<"enter the data which you want to insert at beggining"<<endl;

cin>>data;

L=insertatbeggining(L,data);

printlist(L);

cout<<"after deletion from begginig"<<endl;

L=deletefrombeggining(L);

printlist(L);

cout<<"enter the element which you want to delete\n";

cin>>d;

L=deletenodespecificposition(L,d);

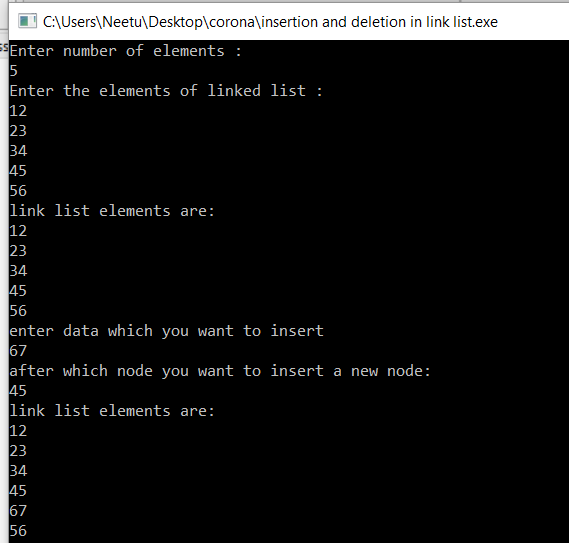
printlist(L);

L=deletefromending(L);

printlist(L);

return 0;

}



**Ques 5:) Evaluate a postfix algebraic expression with the help of stack**

#include<iostream>

#include<stack>

using namespace std;

int evaluatepostfix(string exp)

{

stack<int>s;

for(int i=0;i<exp.length();i++)

{

if(exp[i]>='0' && exp[i]<='9')

s.push(exp[i]-'0');

else

{

int v1=s.top();

s.pop();

int v2=s.top();

s.pop();

if(exp[i]=='+')

s.push(v2+v1);

else if(exp[i]=='-')

s.push(v2-v1);

else if(exp[i]=='\*')

s.push(v2\*v1);

else if(exp[i]=='/')

s.push(v2/v1);

else if(exp[i]=='%')

s.push(v2%v1);

}

}

return s.top();

}

int main()

{

string str;

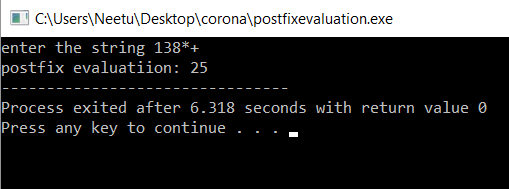
cout<<"enter the string ";

getline(cin,str);

cout<<"postfix evaluatiion: "<<evaluatepostfix(str);

return 0;

}



**Ques 6:) Implement a queue using a linked list and array**.

i)using array

#include<iostream>

using namespace std;

int a[100],n,front =-1,rear=-1;

void insert()

{int value;

if(rear==n-1)

cout<<"queue is full(overflow)\n";

else

{

if(front==-1)

front=0;

cin>>value;

rear++;

a[rear]=value;

}

}

void deleteelement()

{

int value;

if(front==-1||front>rear)

{

cout<<"queue underflow\n";

return ;

}

else

{

cout<<"deleted element from queue is: "<<a[front]<<endl;

front++;

}

}

void display()

{

if(front==-1)

cout<<"queue is empty\n";

else

{

cout<<"queue elements are: ";

for(int i=front;i<=rear;i++)

{

cout<<a[i];

cout<<endl;

}

}

}

int main()

{

cout<<"enter total number of elements in the queue\n";

cin>>n;

cout<<"insert the elements in the queue\n";

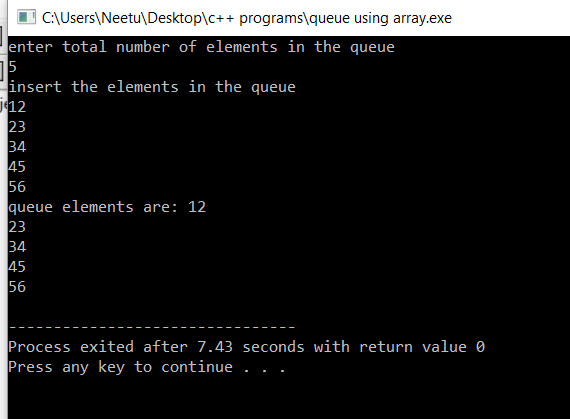
for(int i=0;i<n;i++)

insert();

display();

return 0;

}



ii.)using linked list:

#include<iostream>

#include<stdlib.h>

using namespace std;

struct node

{

int data;

struct node\* next;

};

typedef struct node\* lptr;

lptr temp;

lptr front=NULL;

lptr rear=NULL;

void insertelement(int x)

{

if(rear==NULL)

{

rear=new node();

rear->next=NULL;

rear->data=x;

front=rear;

}

else

{

temp=new node();

rear->next=temp;

temp->data=x;

temp->next=NULL;

rear=temp;

}

}

void show()

{

temp=front;

if(front==NULL && rear==NULL)

cout<<"queue is empty\n";

else

{

cout<<"queue elements are: ";

while(temp!=NULL)

{

cout<<temp->data<<endl;

temp=temp->next;

}

cout<<endl;

}

}

void deleteelement()

{

temp=front;

if(temp==NULL)

cout<<"underflow\n";

else

{

temp=temp->next;

cout<<"delted element is: "<<front->data<<endl;

free(front);

front=temp;

}

}

int main()

{

int n,x;

cout<<"enter the number of values to be entered in the queue\n";

cin>>n;

cout<<"enter the values to be stored in the queue\n";

for(int i=0;i<n;i++)

{ cin>>x;

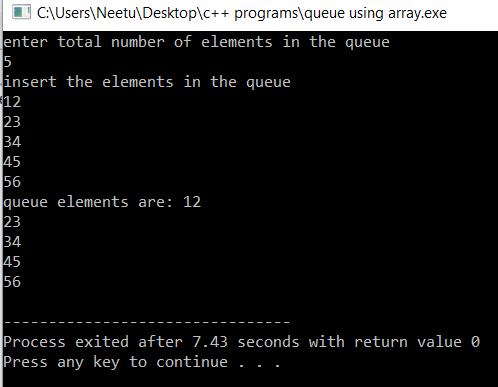
insertelement(x);

}

show();

return 0;

}



Ques **7:) implement a binary tree and any traversal technique**.

#include<iostream>

using namespace std;

struct btnode

{

char data;

struct btnode\*lc;

struct btnode\*rc;

};

typedef struct btnode\*btptr;

btptr create\_node(btptr t,char a[50],int i)

{

if(a[i]!='+')

{

btptr temp=NULL;

temp=new struct btnode;

temp->data=a[i];

temp->lc=NULL;

temp->rc=NULL;

return temp;

}

else

return NULL;

}

btptr create\_tree(btptr t,char a[50],int i,int n)

{

if(i>=n)

return t;

t=create\_node(t,a,i);

if(t==NULL)

return NULL;

i++;

t->lc=create\_tree(t->lc,a,i,n);

i++;

t->rc=create\_tree(t->rc,a,i,n);

}

void in(btptr t)

{

if(t)

{

in(t->lc);

cout<<t->data<<endl;

in(t->rc);

}

}

int main()

{

btptr root=NULL;

int n;

char a[50];

cout<<"enter the total number of nodes"<<endl;

cin>>n;

cout<<"enter the elements"<<endl;

for(int i=0;i<n;i++)

cin>>a[i];

int j=0;

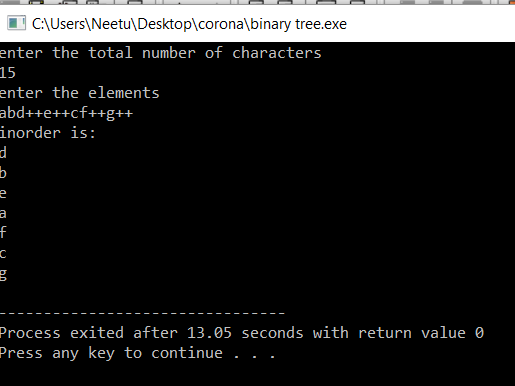
root=create\_tree(root,a,j,n);

cout<<"inorder is: "<<endl;

in(root);

return 0;

}



**Ques 8:) implement a binary search tree and insert and delete in binary search tree.**

#include<iostream>

using namespace std;

struct BSTnode

{

int data;

struct BSTnode \*lchild, \*rchild;

};

struct BSTnode \*getnewnode(int data)

{

struct BSTnode\* temp=new BSTnode();

temp->data=data;

temp->lchild=NULL;

temp->rchild=NULL;

return temp;

}

struct BSTnode\* insertelement(struct BSTnode\* root,int data)

{

if(root==NULL)

root=getnewnode(data);

else if(data<=root->data)

root->lchild=insertelement(root->lchild,data);

else

root->rchild=insertelement(root->rchild,data);

return root;

}

BSTnode\* minimumkey(BSTnode\* curr)

{

while(curr->lchild!=NULL)

curr=curr->lchild;

return curr;

}

void searchkey(BSTnode\* curr,int key,BSTnode\* parent)

{

while(curr!=NULL && curr->data!=key)

{

parent=curr;

if(key<curr->data)

curr=curr->lchild;

else

curr=curr->rchild;

}

}

void deletenode(BSTnode\* root,int key)

{

BSTnode\*parent=NULL;

BSTnode \*curr=root;

searchkey(curr,key,parent);

if(curr==NULL)

return;

//case 1: node with 0 child

if(curr->lchild==NULL &&curr->rchild==NULL)

{

if(curr!=root)

{

if(parent->lchild==curr)

parent->lchild==NULL;

else

parent->rchild==NULL;

}

else

root=NULL;

free(curr);

}

//case 2: node with 2 child

else if(curr->lchild && curr->rchild)

{

BSTnode\*successor=minimumkey(curr->rchild);

int val=successor->data;

deletenode(root,successor->data);

curr->data=val;

}

//case 3: node with 1 child

else

{

BSTnode\* child=(curr->lchild)?curr->lchild:curr->rchild;

if(curr!=root)

{

if(curr==parent->lchild)

parent->lchild=child;

else

parent->rchild=child;

}

else

root=child;

free(curr);

}

}

void inorder(struct BSTnode\* t)

{

if(t)

{

inorder(t->lchild);

cout<<t->data<<endl;

inorder(t->rchild);

}

}

int main()

{

int n,data;

struct BSTnode\* root=NULL;

cout<<"enter number of nodes in tree";

cin>>n;

cout<<"enter data\n";

for(int i=0;i<n;i++)

{ cin>>data;

root=insertelement(root,data);

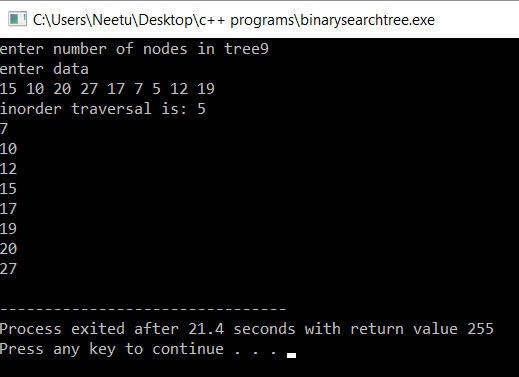
}

cout<<"inorder traversal is: ";

inorder(root);

return 0;

}



**Ques 9:) Implement a max priority queue using max heap.**

#include <iostream>

using namespace std;

int heap\_size = 0;

int n=20;

const int INF = 100000;

int get\_right\_child(int A[], int index)

{

if((((2\*index)+1) < n) && (index >= 1))

return (2\*index)+1;

return -1;

}

int get\_left\_child(int A[], int index)

{

if(((2\*index) < n) && (index >= 1))

return 2\*index;

return -1;

}

int get\_parent(int A[], int index)

{

if ((index > 1) && (index < n))

{

return index/2;

}

return -1;

}

void swap( int \*a, int \*b )

{

int temp;

temp = \*a;

\*a = \*b;

\*b = temp;

}

void max\_heapify(int A[], int index)

{

int left\_child\_index = get\_left\_child(A, index);

int right\_child\_index = get\_right\_child(A, index);

int largest = index;

if ((left\_child\_index <= heap\_size) && (left\_child\_index>0))

{

if (A[left\_child\_index] > A[largest])

{

largest = left\_child\_index;

}

}

if ((right\_child\_index <= heap\_size && (right\_child\_index>0)))

{

if (A[right\_child\_index] > A[largest])

{

largest = right\_child\_index;

}

}

// largest is not the node, node is not a heap

if (largest != index) {

swap(&A[index], &A[largest]);

max\_heapify(A, largest);

}

}

void build\_max\_heap(int A[])

{

int i;

for(i=heap\_size/2; i>=1; i--)

{

max\_heapify(A, i);

}

}

int extract\_max(int A[])

{

int max = A[1];

A[1] = A[heap\_size];

heap\_size--;

max\_heapify(A, 1);

return max;

}

void insertion(int A[], int index, int key)

{

A[index] = key;

while((index>1) && (A[get\_parent(A, index)] < A[index]))

{

swap(&A[index], &A[get\_parent(A, index)]);

index = get\_parent(A, index);

}

}

void insert(int A[], int key)

{

heap\_size++;

A[heap\_size] = -1\*INF;

insertion(A, heap\_size, key);

}

void print\_heap(int A[])

{

int i;

for(i=1; i<=heap\_size; i++)

{

cout<<A[i]<<" ";

}

cout<<endl;

}

int main()

{

int A[n];

int m,v;

cout<<"number of elements in queue: ";

cin>>m;

for(int i=0;i<m;i++)

{

cin>>v;

insert(A,v);

}

cout<<" max heap is: ";

print\_heap(A);

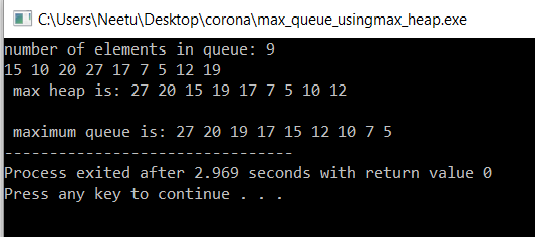
cout<<"\n maximum queue is: ";

for(int i=0;i<m;i++)

cout<<extract\_max(A)<<" ";

return 0;

}

****

**Ques 10:) Implement a graph and \_nd transpose of a graph where Transpose of a directed graph G is another directed graph on the same set of vertices with all of the edges reversed compared to the orientation of the corresponding edges in G. That is, if G contains an edge (u, v) then the converse/transpose/reverse of G contains an edge (v, u) and vice versa. Implement it with the help of adjacency list and adjacency matrix**

#include<iostream>

using namespace std;

int main()

{

int i,v,e,j,v1,v2,g[20][20],a,indegree[50]={0},outdegree[50]={0};

cout<<"enter the number of vertices end edges of the graph: ";

cin>>v>>e;

for(i=1;i<=v;i++)

for(j=1;j<=v;j++)

g[i][j]=0;

cout<<"\nenter the vertices v1 and v2 between which edge exists";

for(i=1;i<=e;i++)

{

cin>>v1>>v2;

g[v1][v2]=1;

}

for(i=1;i<=v;i++)

{

for(j=1;j<=v;j++)

cout<<g[i][j]<<" ";

cout<<endl;

}

for(i=1;i<=e;i++)

{

g[v2][v1]=g[v1][v2];

}

Cout<<”transpose of graph is:”;

for(i=1;i<=v;i++)

{

for(j=1;j<=v;j++)

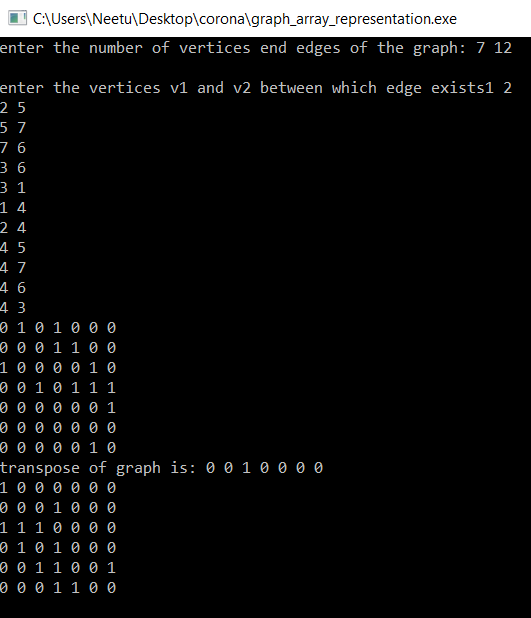
cout<<g[j][i]<<" ";

cout<<endl;

}

return 0;

}



**Ques 11:) Implement quick sort,merge sort,insertion sort,selection sort.**

**I:) Quick sort:**

#include<iostream>

using namespace std;

void swap(int\* a, int\* b)

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int partition (int arr[], int l, int h)

{

int pivot = arr[h];

int i = (l- 1);

for (int j = l; j <= h- 1; j++)

{

if (arr[j] <= pivot)

{

i++;

swap(&arr[i], &arr[j]);

}

}

swap(&arr[i + 1], &arr[h]);

return (i + 1);

}

void quickSort(int arr[], int l, int h)

{

if (l<h)

{

int pi = partition(arr,l,h);

quickSort(arr, l, pi - 1);

quickSort(arr, pi +1,h);

}

}

void printArray(int arr[], int size)

{

int i;

for (i=0; i < size; i++)

cout<<arr[i]<<endl;

}

int main()

{

int arr[20],n;

cout<<"enter size of an array ";

cin>>n;

cout<<"enter elements of an array ";

for(int i=0;i<n;i++)

cin>>arr[i];

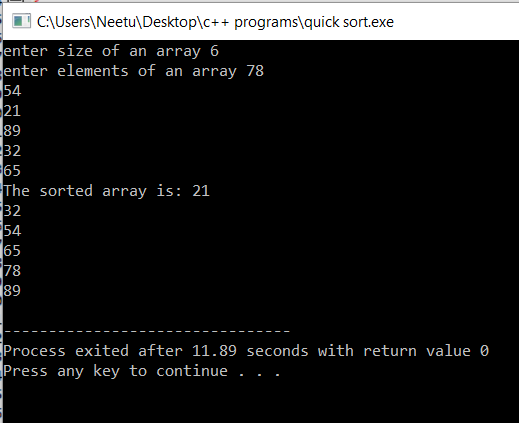
quickSort(arr, 0, n-1);

printf("The sorted array is: ");

printArray(arr, n);

return 0;

}



**ii) Merge sort**:

#include<iostream>

using namespace std;

int merge(int a[],int p,int q,int r)

{

int n1,n2;

int i,j;

n1=q-p+1;

n2=r-q;

int L[n1+1],R[n2+1];

for(i=1;i<=n1;i++)

L[i]=a[i+p-1];

for(j=1;j<=n2;j++)

R[j]=a[j+q];

i=1;j=1;

L[n1+1]= {10000};

R[n2+1]={10000};

for(int k=p;k<=r;k++)

{

if(L[i]<=R[j])

{

a[k]=L[i];

i++;

}

else

{

a[k]=R[j];

j++;

}

}

}

int merge\_sort(int a[],int p,int r)

{

if(p<r)

{

int q=(p+r)/2;

merge\_sort(a,p,q);

merge\_sort(a,q+1,r);

merge(a,p,q,r);

}

}

int main()

{

int n,a[200];

cout<<"enter total number of elements in the array"<<endl;

cin>>n;

cout<<"enter elements of the array: "<<endl;

for(int i=0;i<n;i++)

cin>>a[i];

cout<<"sorted ARRAY IS:"<<endl;

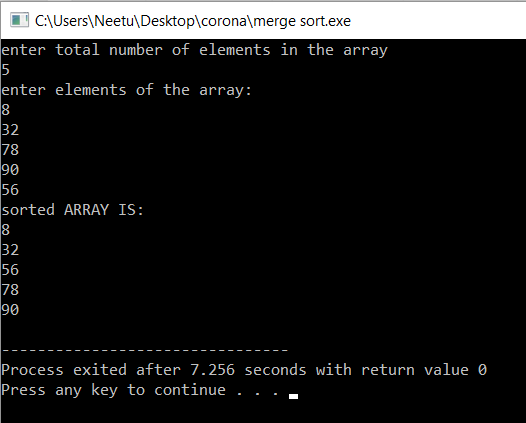
merge\_sort(a,1,n);

for(int i=0;i<n;i++)

cout<<a[i]<<endl;

return 0;

}



**iii) selection sort**:

#include<iostream>

using namespace std;

void swap(int \*a,int \*b)

{

int temp=\*a;

\*a=\*b;

\*b=temp;

}

void selection\_sort(int a[],int n)

{

int i,j,min,temp;

for(i=0;i<n-1;i++)

{

min=i;

for(j=i+1;j<n;j++)

{

if(a[j]<a[min])

min=j;

}

swap(&a[min],&a[i]);

cout<<a[i]<<" ";

}

}

int main()

{

int\*a=NULL,n;

a=new int[n];

cout<<"enter the number of elements of an array"<<endl;

cin>>n;

cout<<"enter the elements of an array"<<endl;

for(int i=0;i<n;i++)

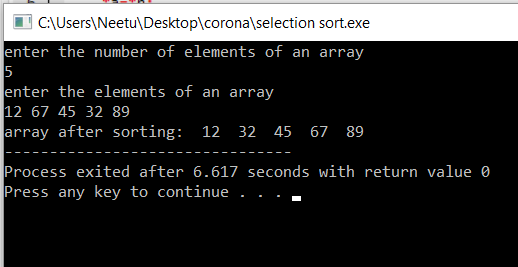
cin>>a[i];

cout<<"array after sorting: "<<" ";

selection\_sort(a,n);

return 0;

}



iv**)Insertion sort:**

#include<iostream>

using namespace std;

void insertionsort(int a[],int n)

{

int j;

for(int i=0;i<n;i++)

{

int k;

k=a[i];

j=i-1;

while(j>=0 && a[j]>k)

{

a[j+1]=a[j];

j=j-1;

}

a[j+1]=k;

}

}

int main()

{

int n,\*a=NULL;

cout<<"enter number of elements in array"<<endl;

cin>>n;

a=new int [n];

cout<<"enter elements in the array"<<endl;

for(int i=0;i<n;i++)

cin>>a[i];

insertionsort(a,n);

cout<<"array after sorting: "<<endl;

for(int i=0;i<n;i++)

cout<<a[i]<<endl;

return 0;

}

