**Q. Write a program to implement linear search.**

**Code:**

#include<iostream.h>

#include<conio.h>

int main(){

int i,x;

int A[10];

clrscr();

cout<<"Enter 10 numbers";

for(i=0;i<10;i++)

cin>>A[i];

cout<<"Enter the number to search";

cin>>x;

for(i=0;i<10;i++)

if(A[i]==x){

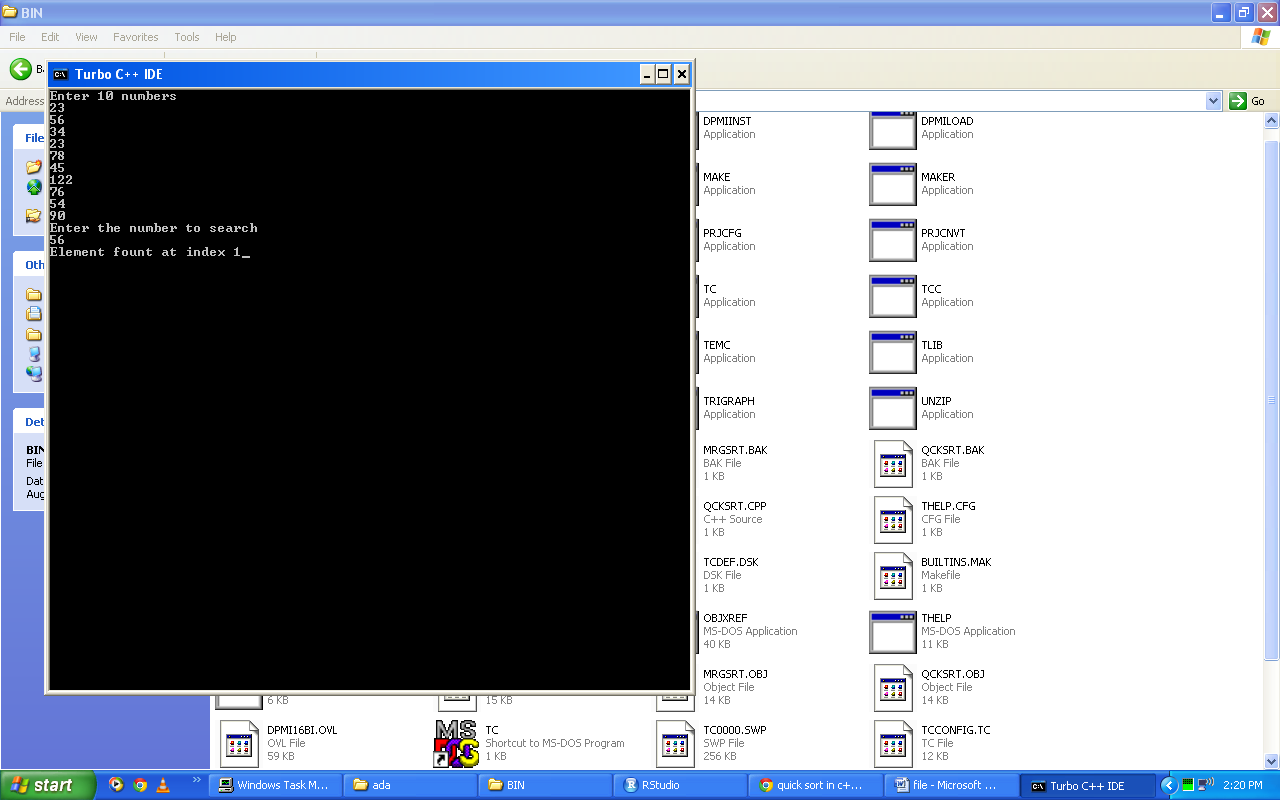
cout<<"Element fount at index "<<i;

return 0; }

cout<<Element not found";

return 0;}

**Output:**

****

**Q. Write a program to compare the complexity of insertion, selection and bubble sort.**

**Code:**

#include<iostream.h>

#include<conio.h>

#include<time.h>

void main(){

int A[10],B[10],C[10],i,temp,j;

clrscr();

cout<<"Enter 10 elements\n";

for(i=0;i<10;i++){

cin>>A[i];

B[i]=A[i];

C[i]=A[i];

}

cout<<"Insertion Sort:\n";

clock\_t start=clock();

for(long k=0;k<10000000;k++){

for(i=0;i<10;i++){

temp=A[i];

j=i-1;

while(A[j]>temp && j>=0){

A[j+1]=A[j];

j--;

}

A[j+1]=temp;

}

}

cout<<"Time Complexity:\n";

cout<<((clock()-start)/(double)CLK\_TCK);

cout<<"\n\n Bubble Sort:\n";

start=clock();

for(k=0;k<10000000;k++){

for(i=0;i<9;i++){

for(j=0;j<10;j++){

if(B[j]>B[j+1]){

temp=B[j];

B[j]=B[j+1];

B[j+1]=temp;

}

}

}

}

cout<<"Time Complexity:\n";

cout<<((clock()-start)/(double)CLK\_TCK);

cout<<"\n\n Selection Sort:\n";

start=clock();

for(k=0;k<10000000;k++){

for(i=0;i<10;i++){

int min=C[i];

int pos=i;

for(j=i+1;j<10;j++){

if(C[j]<min){

min=C[j];

pos=j;

}

}

temp=C[i];

C[i]=C[pos];

C[pos]=temp;

}

}

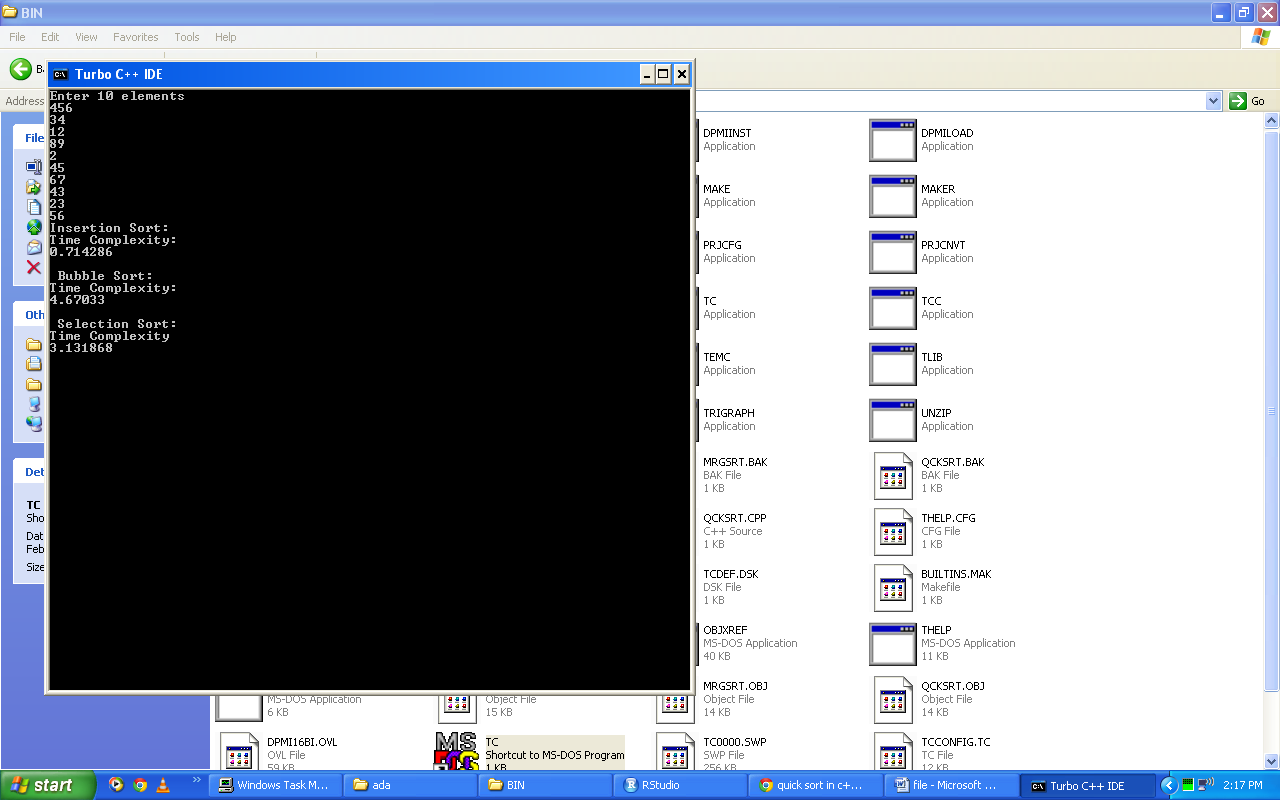
cout<<"Time Complexity\n";

cout<<((clock()-start)/(double)CLK\_TCK);

getch();

}

**Output:**

****

**Q. Write a program for implementing merge sort**

**Code:**

#include<iostream.h>

#include<conio.h>

#include<time.h>

#include<math.h>

int a[5];

void merge(int p, int q, int m){

int n1=m-p+1;

int n2=q-m;

int b[5],c[5],i;

for(i=0;i<n1;i++)

b[i]=a[(p+i)];

for(i=0;i<n2;i++)

c[i]=a[(m+1+i)];

int i\_b=0,i\_c=0,q\_b=n1,q\_c=n2,k=p;

while(i\_b<q\_b && i\_c<q\_c){

if(b[i\_b]<=c[i\_c]){

a[k]=b[i\_b];

i\_b++; }

else{

a[k]=c[i\_c];

i\_c++; }

k++; }

while(i\_b<q\_b){

a[k]=b[i\_b]; i\_b++; k++; }

while(i\_c<q\_c){

a[k]=c[i\_c]; i\_c++; k++; }}

void merge\_sort(int p, int q){

int m;

if(p<q){

m=floor((p+q)/2);

merge\_sort(p,m);

merge\_sort(m+1,q);

merge(p,q,m); }}

void main(){

int i;

clrscr();

cout<<"Enter 5 numbers.\n";

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

cin>>a[i];

clock\_t start=clock();

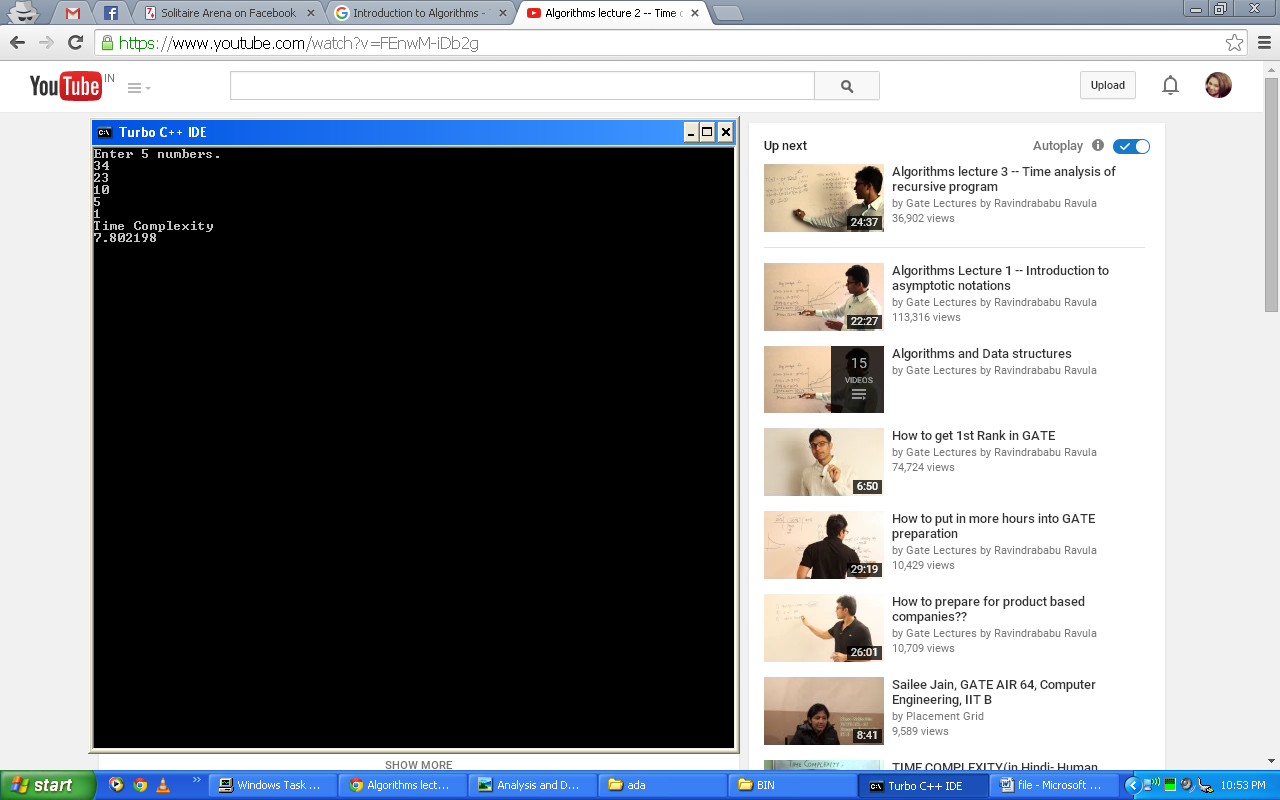
for(long k=0;k<10000000;k++)

merge\_sort(0,4);

cout<<"Time Complexity\n";

cout<<((clock()-start)/(double)CLOCKS\_PER\_SEC);}

**Output:**

****

**Q. Write a program to implement quick sort.**

**Code:**

#include<iostream.h>

#include<conio.h>

#include<time.h>

int a[5];

int partition(int p, int r){

int x=a[r];

int i=p-1;

int temp;

for(int j=p;j<r;j++){

if(a[j]<=x){

i++;

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

temp=a[i+1];

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

a[r]=temp;

return i+1;

}

void quick\_sort(int p,int r){

int q;

if(p<r){

q=partition(p,r);

quick\_sort(p,q-1);

quick\_sort(q+1,r);

}

}

void main(){

int i;

clrscr();

cout<<"Enter 5 numbers\n";

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

cin>>a[i];

clock\_t start=clock();

for(long k=0;k<10000000;k++)

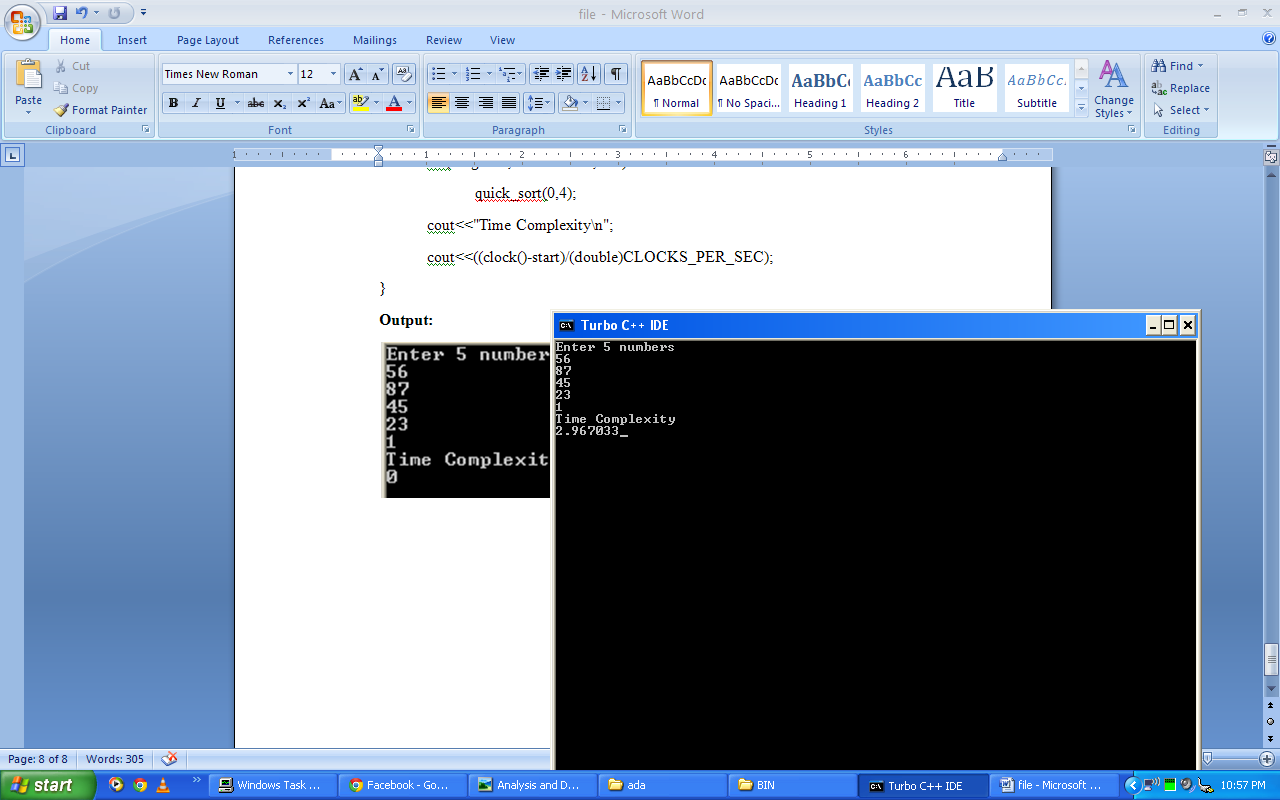
quick\_sort(0,4);

cout<<"Time Complexity\n";

cout<<((clock()-start)/(double)CLOCKS\_PER\_SEC);

}

**Output:**

****

**Q. Write a program to implement Strassen Matrix Multiplication.**

**Code:**

#include<iostream.h>

#include<conio.h>

void main(){

int a[2][2],b[2][2],c[2][2],i,j;

int m1,m2,m3,m4,m5,m6,m7;

clrscr();

cout<<"Enter the 4 elements of first matrix:\n";

for(i=0;i<2;i++)

for(j=0;j<2;j++)

cin>>a[i][j];

cout<<"Enter the 4 elements of second matrix:\n";

for(i=0;i<2;i++)

for(j=0;j<2;j++)

cin>>b[i][j];

cout<<"\nThe first matrix is\n";

for(i=0;i<2;i++){

cout<<"\n";

for(j=0;j<2;j++)

cout<<a[i][j]<<"\t";

} cout<<"\nThe second matrix is\n";

for(i=0;i<2;i++){

cout<<"\n";

for(j=0;j<2;j++)

cout<<b[i][j]<<"\t";

}

m1= (a[0][0]+a[1][1])\*(b[0][0]+b[1][1]);

m2= (a[1][0]+a[1][1])\*b[0][0];

m3= a[0][0]\*(b[0][1]-b[1][1]);

m4= a[1][1]\*(b[1][0]-b[0][0]);

m5= (a[0][0]+a[0][1])\*b[1][1];

m6= (a[1][0]-a[0][0])\*(b[0][0]+b[0][1]);

m7= (a[0][1]-a[1][1])\*(b[1][0]+b[1][1]);

c[0][0]=m1+m4-m5+m7;

c[0][1]=m3+m5;

c[1][0]=m2+m4;

c[1][1]=m1-m2+m3+m6;

cout<<"\nMatrix Product:\n";

for(i=0;i<2;i++){

cout<<"\n";

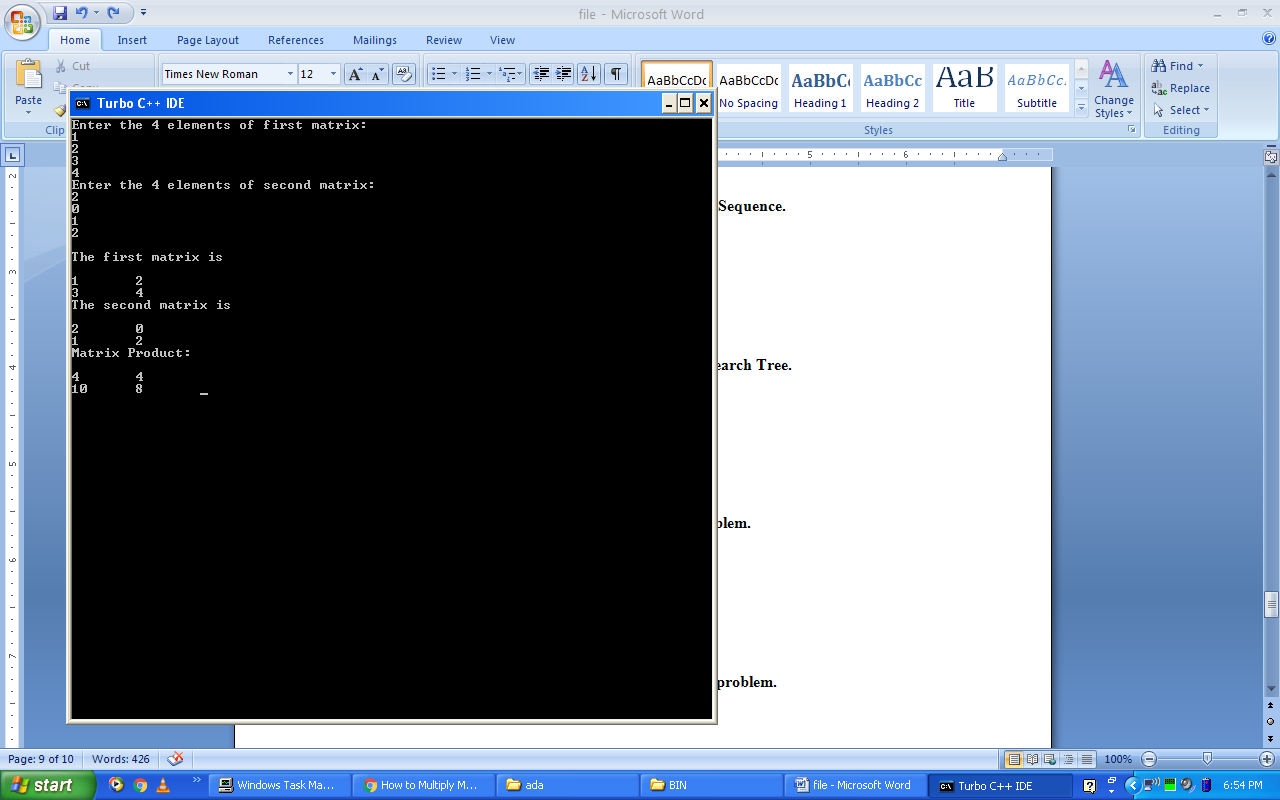
for(j=0;j<2;j++)

cout<<c[i][j]<<"\t";

} getch();

}

**Output:**

****

**Q. Write a program to implement Longest Common Sequence.**

**Code:**

#include<iostream.h>#include<string.h>#include<conio.h>

int i,j,m,n,c[20][20];

char x[20],y[20],b[20][20];

void print(int i,int j){

if(i==0 || j==0)

return;

if(b[i][j]=='c'){

print(i-1,j-1);

cout<<x[i-1];

}

else if(b[i][j]=='u')

print(i-1,j);

else

print(i,j-1);

}

void lcs(){

m=strlen(x);

n=strlen(y);

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

c[i][0]=0;

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

c[0][i]=0;

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

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

if(x[i-1]==y[j-1]){

c[i][j]=c[i-1][j-1]+1;

b[i][j]='c';

}

else if(c[i-1][j]>=c[i][j-1]){

c[i][j]=c[i-1][j];

b[i][j]='u';

}

else{

c[i][j]=c[i][j-1];

b[i][j]='l';

}

}

}

void main(){

clrscr();

cout<<"Enter 1st sequence:";

cin>>x;

cout<<"Enter 2nd sequence:";

cin>>y;

cout<<"\nThe Longest Common Subsequence is ";

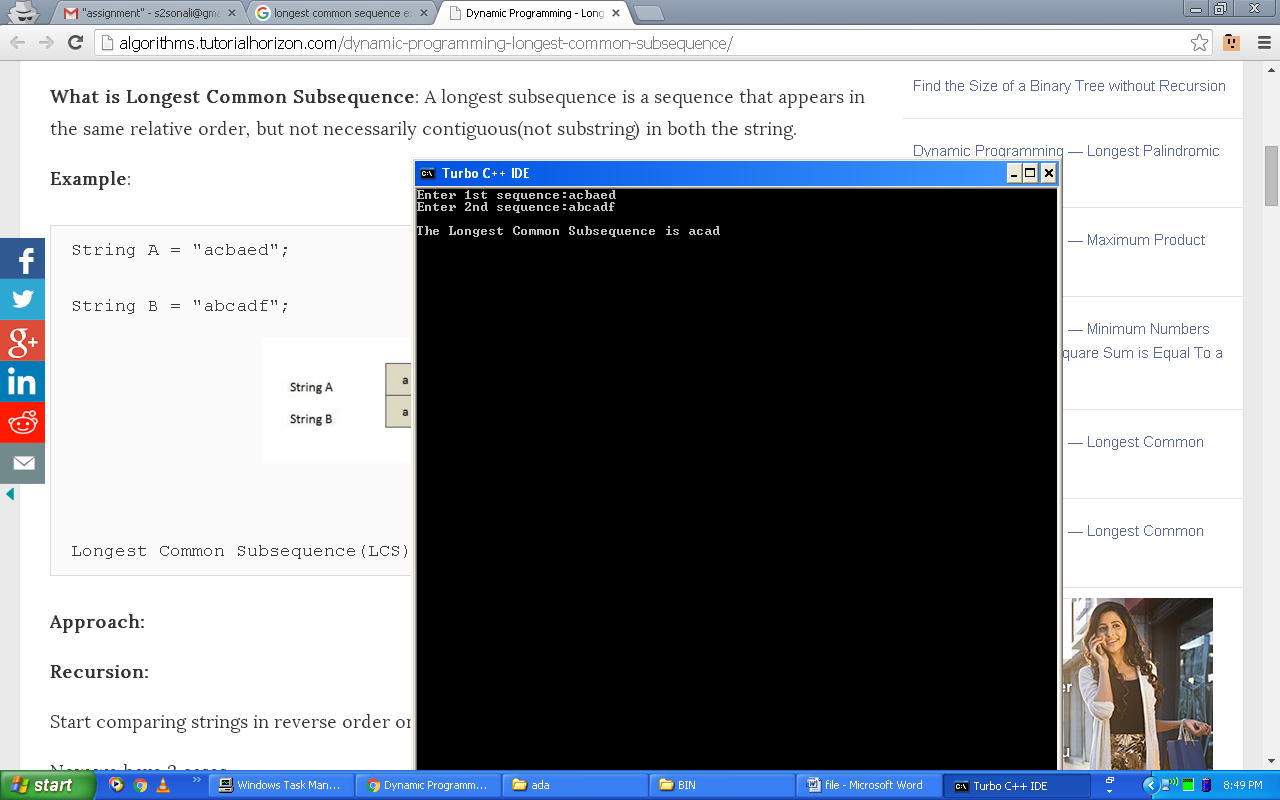
lcs();

print(m,n);

getch();

}

**Output:**

****

**Q. Write a program to implement Optimal Binary Search Tree.**

**Code:**

#include<iostream.h>

#include<conio.h>

#define MAX 10

void main()

{

double w[MAX][MAX], c[MAX][MAX], p[MAX], q[MAX];

double min, min1;

int temp=0,r[MAX][MAX],i,j,k,b,root,n;

clrscr();

cout<<"Enter the number of elements:\t";

cin>>n;

cout<<"\n";

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

cout<<"Enter the Successful Search Probability of Element "<<i<<":\t";

cin>>p[i];

}

cout<<"\n";

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

cout<<"Enter the Unsuccessful Search Probability of Element "<<i<<":\t";

cin>>q[i];

}

cout<<"W\t\tC\t\tR\n";

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

for(j=0; j <= n; j++){

if(i == j){

w[i][j] = q[i];

c[i][j] = 0;

r[i][j] = 0;

cout<<"W["<<i<<"]["<<j<<"]: "<<w[i][j]<<"\tC["<<i<<"]["<<j<<"]: "<<c[i][j]<<"\tR["<<i<<"]["<<j<<"]: "<<r[i][j]<<"\n";

}

}

}

cout<<"\n";

for(b=0; b < n; b++){

for(i=0,j=b+1; j < n+1 && i < n+1; j++,i++){

if(i!=j && i < j){

w[i][j] = p[j] + q[j] + w[i][j-1];

min = 30000;

for(k = i+1; k <= j; k++){

min1 = c[i][k-1] + c[k][j] + w[i][j];

if(min > min1){

min = min1;

temp = k;

}

}

c[i][j] = min;

r[i][j] = temp;

}

cout<<"W["<<i<<"]["<<j<<"]: "<<w[i][j]<<"\tC["<<i<<"]["<<j<<"]: "<<c[i][j]<<"\tR["<<i<<"]["<<j<<"]: "<<r[i][j]<<"\n";

}

cout<<"\n";

}

cout<<"Minimum cost = "<<c[0][n]<<"\n";

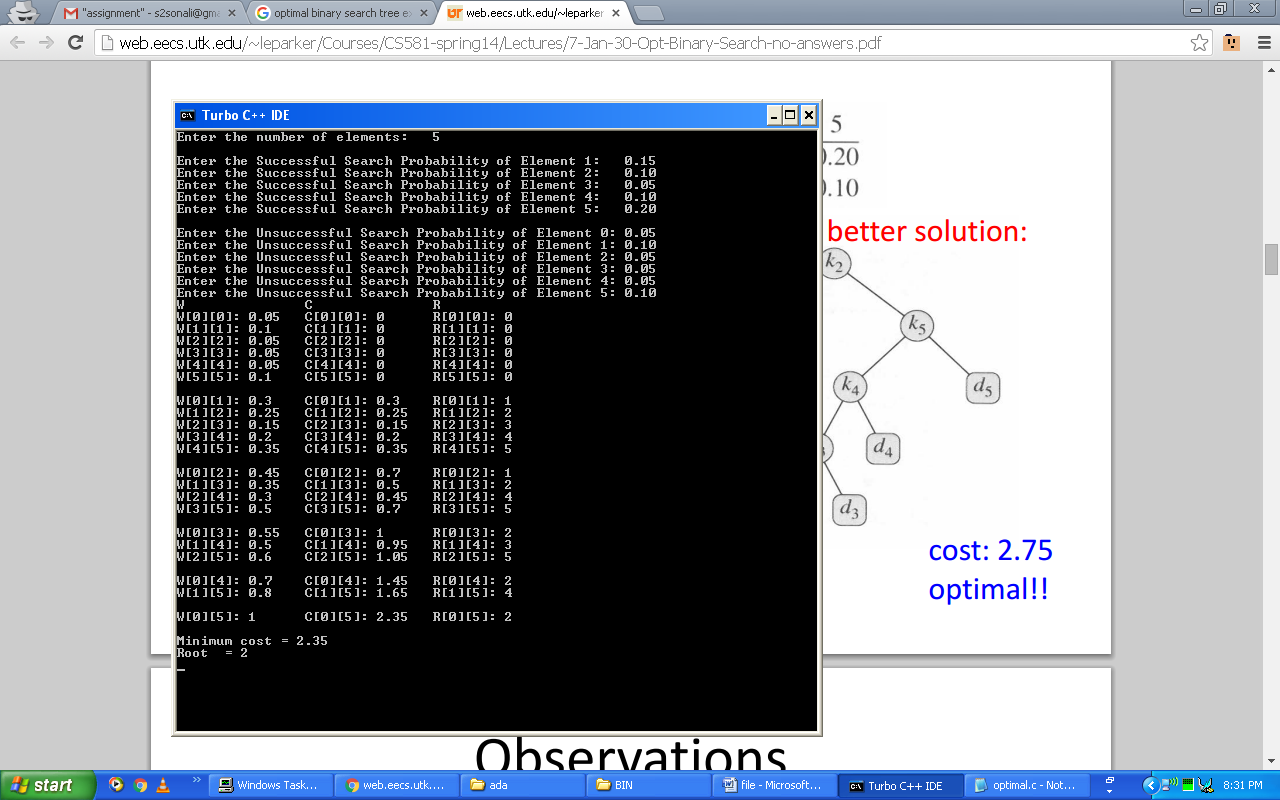
root = r[0][n];

cout<<"Root = "<<root<<"\n";

getch();

}

**Output:**

****

**Q. Write a program to implement 0/1 Knapsack problem.**

**Code:**

#include<iostream.h>

#include<conio.h>

int sum=0;

int max(int a,int b){

if(a>b)

return a;

else

return b;

}

void knapsack(int m,int n,int w[],int p[]){

int v[100][200],x[10],i,j;

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

v[0][i]=0;

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

for(j=0;j<=m;j++){

if(j>=w[i])

v[i][j]=max(v[i-1][j],v[i-1][j-w[i]]+p[i]);

else

v[i][j]=v[i-1][j];

}

}

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

x[i]=0;

i=n;

j=m;

while(i>0 && j>0){

if(v[i][j]!=v[i-1][j]){

x[i]=1;

j=j-w[i];

}

i--;

}

cout<<"\nThe Optimal Set of Weights is:\n";

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

if(x[i]==1){

cout<<"X"<<i<<"=1\t";

sum=sum+p[i];

}

else

cout<<"X"<<i<<"=0\t";

}

cout<<"\nTotal profit = "<<sum;

}

void main(){

int w[10],p[10],i,m,n;

clrscr();

cout<<"\t0/1 Knapsack Problem\n\n";

cout<<"Enter The Number of Items:\t";

cin>>n;

cout<<"Enter the Weights of the Items:\n";

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

cin>>w[i];

cout<<"Enter the Profits of the Items:\n";

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

cin>>p[i];

cout<<"Enter the Capacity of Knapsack: ";

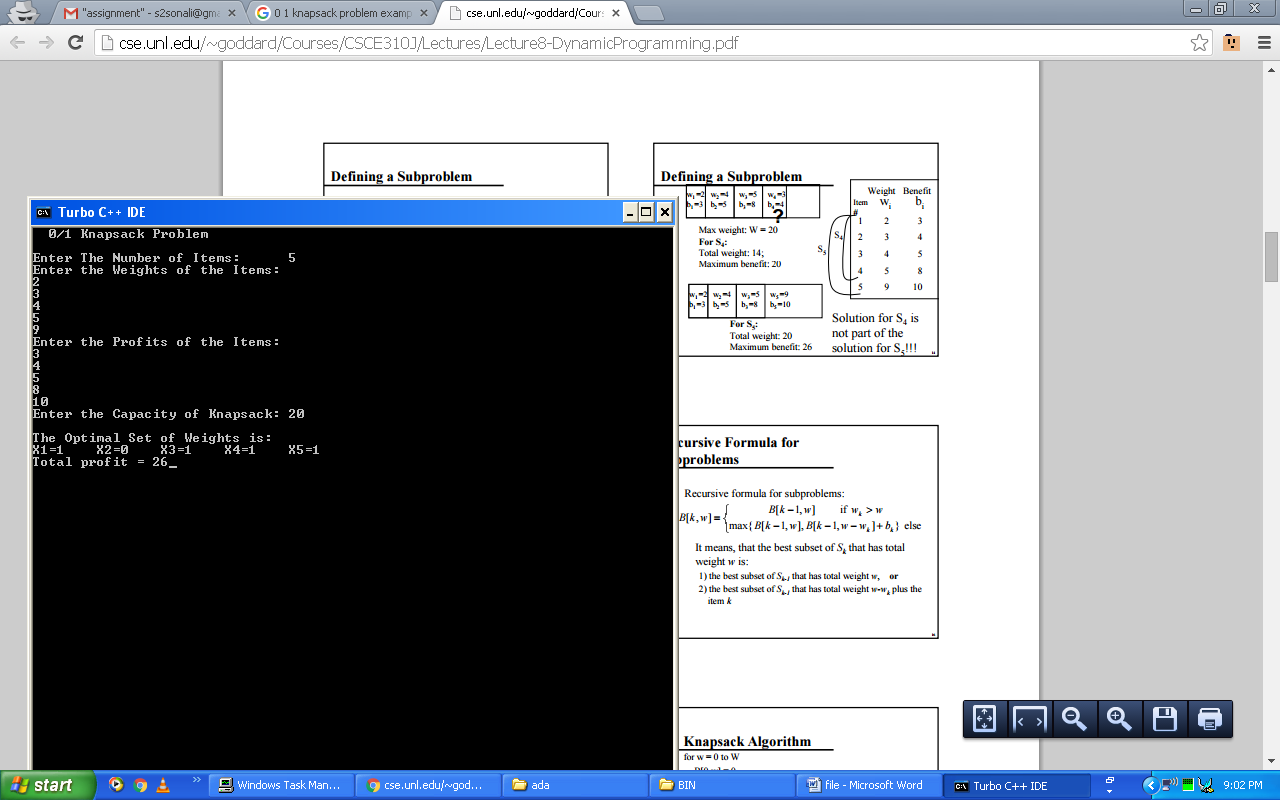
cin>>m;

knapsack(m,n,w,p);

getch();

}

**Output:**

****

**Q. Write a program to implement Activity Selection problem.**

**Code:**

#include<iostream.h>

#include<conio.h>

void printMaxAct(int s[],int f[],int n){

int i,j;

cout<<"Following activities are selected\n";

i=0;

cout<<i;

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

if(s[j]>=f[i]){

cout<<j;

i=j;

}

}

}

void main(){

clrscr();

int s[]={1,3,0,5,8,5};

int f[]={2,4,6,7,9,9};

int n=sizeof(s)/sizeof(s[0]);

printMaxAct(s,f,n);

getch();

}

**Output:**

****

**Q. Write a program to implement Naïve String matching algorithm.**

**Code:**

#include<conio.h>

#include<iostream.h>

#include<stdio.h>

#include<string.h>

void naive\_string(char t[20], char p[20], int n, int m)

{

int s,a,j,ctr=0;

for(s=0;s<=n-m;s++){

j=1;

while(j<=m-1 && t[s+j]==p[j])

j++;

if(j>=m){

cout<<"\nPattern is found at position "<<s+1;

ctr=1;

}

}

if(ctr==0)

cout<<"Pattern is not found\n";

}

void main(){

char t[20],p[10];

int n,m,z,i;

clrscr();

cout<<"\tString matching by naive method\n";

cout<<"\nEnter text\n";

gets(t);

cout<<"\nEnter pattern\n";

gets(p);

n=strlen(t);

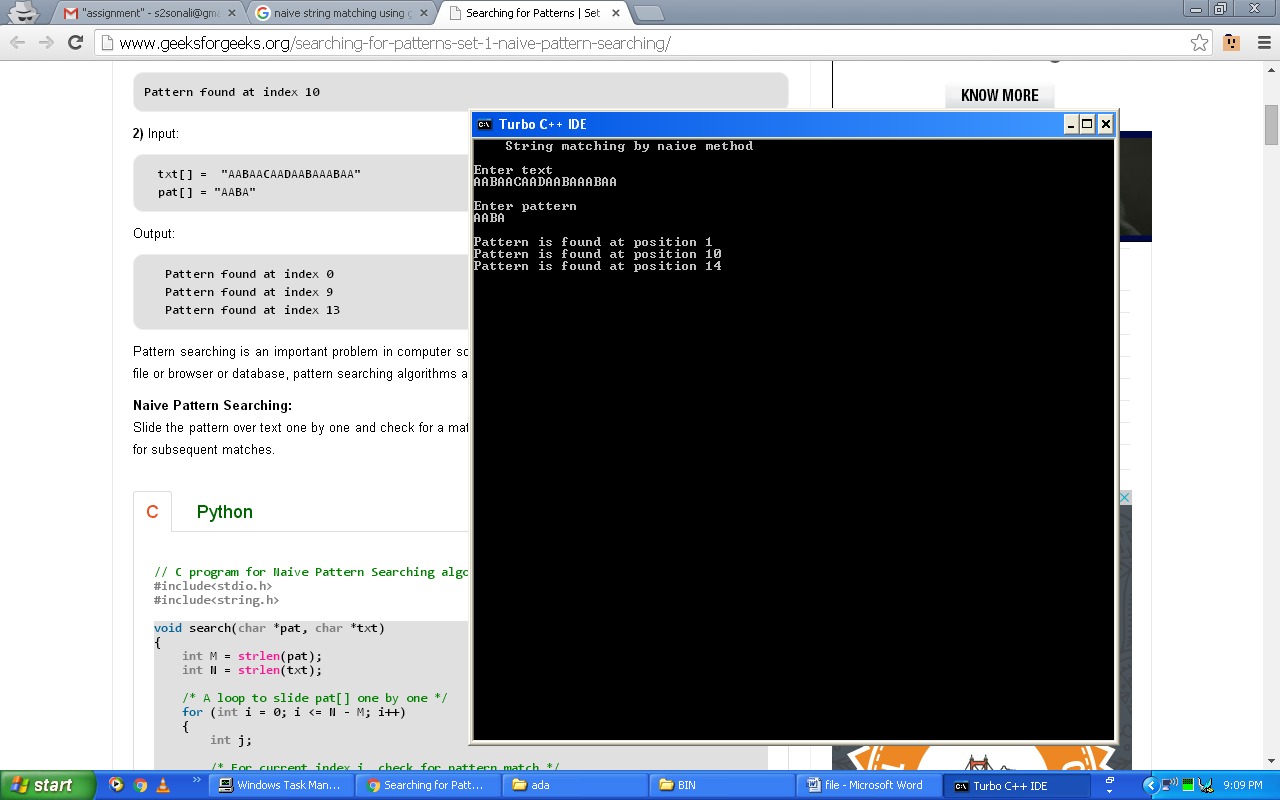
m=strlen(p);

naive\_string(t,p,n,m);

getch();

}

**Output:**

****

**Q. Write a program to implement Breadth First Search.**

**Code:**

#include<iostream.h>

#include<conio.h>

char que[20];

int front=0,rear=0,n;

char arr[20];

int bfs(int );

char ajMat[20][20];

char b[20];

void display();

int p=0;

void main(){

clrscr();

char v;

int i;

cout<<"Enter the number of nodes in the graph\n";

cin>>n;

cout<<"Enter the value of node of graph\n";

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

cin>>b[i];

cout<<"Enter the value in adjacency matrix in form of 'y' or 'n'\n";

cout<<"\nIf there exists an edge between two vertices than 'y' otherwise 'n'\n";

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

for(int j=0;j<n;j++){

cout<<b[i]<<"->"<<b[j]<<": ";

char v;

cin>>v;

ajMat[i][j]=v;

}

cout<<"\n\n";

}

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

bfs(i);

display();

getch();

}

void display(){

cout<<" BFS of Graph: ";

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

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

}

void insert(char val){

que[front]=val;

front++;

}

char del(){

rear=rear+1;

return que[rear-1];

}

int unVisit(char val){

for(int i=0;i<front;i++){

if(val==que[i])

return 0;

}

return 1;

}

void bfs(int i){

char m;

if(front==0)

insert(b[i]);

for(int j=0;j<n;j++){

if(ajMat[i][j]=='y'){

if(unVisit(b[j]))

insert(b[j]);

} }

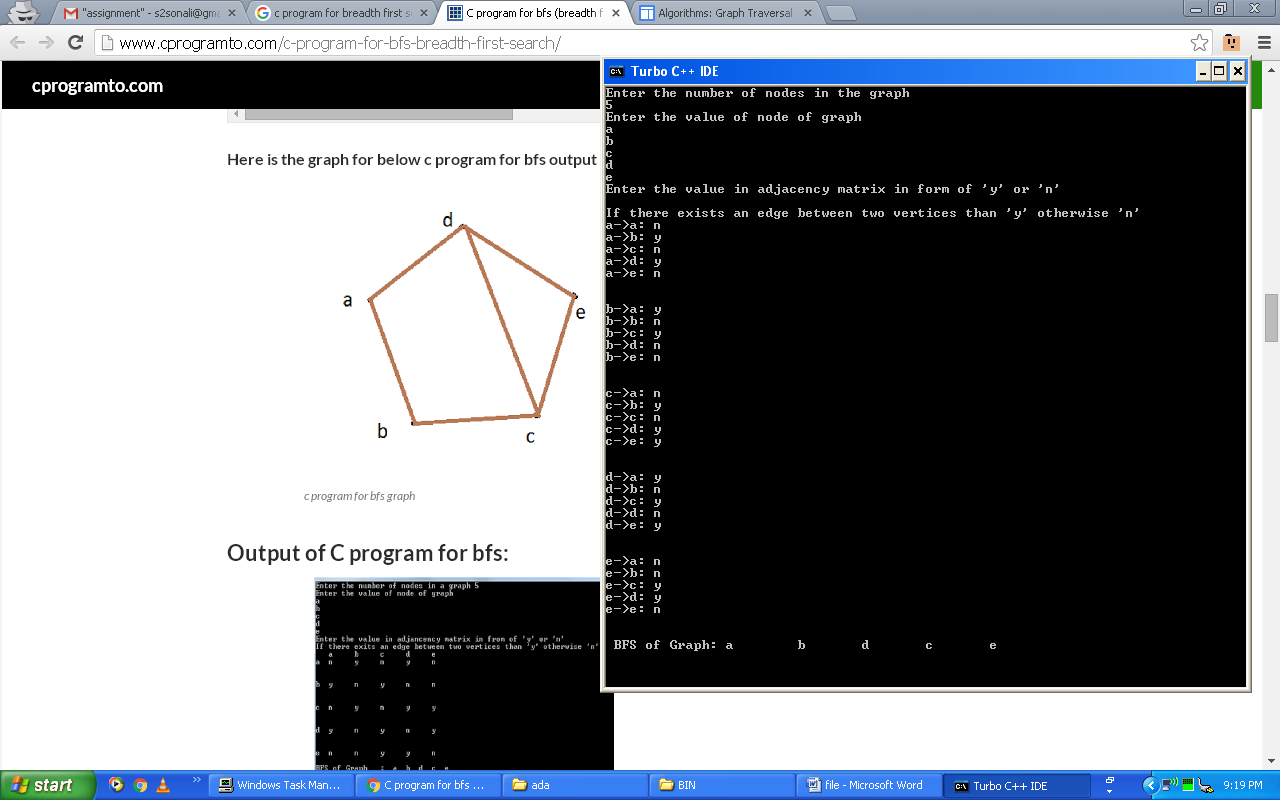
m=del();

arr[p]=m;

p++;

}

**Output:**

****

**Q. Write a program to implement Depth First Search.**

**Code:**

#include<iostream.h>

#include<conio.h>

char stack[20];

int top=-1,n;

char arr[20];

char dfs(int );

char ajMat[20][20];

char b[20];

void display();

int p=0;

void main(){

clrscr();

int i;

char v;

int l=0;

cout<<"Enter the number of nodes in a graph\n";

cin>>n;

cout<<"Enter the value of node of graph\n";

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

cin>>b[i];

char k=b[0];

cout<<"Enter the value in adjancency matrix in form of 'y' or 'n'\n";

cout<<"\nIf there is an edge between the two vertices then enter 'y' or 'n'\n";

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

for(int j=0;j<n;j++){

cout<<b[i]<<"->"<<b[j]<<": ";

char v;

cin>>v;

ajMat[i][j]=v;

}

cout<<"\n\n";

}

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

l=0;

while(k!=b[l])

l++;

k=dfs(l);

}

display();

getch();

}

void display(){

cout<<" DFS of Graph: ";

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

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

}

void push(char val){

top=top+1;

stack[top]=val;

}

char pop(){

return stack[top];

}

int unVisit(char val){

int i;

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

if(val==arr[i])

return 0;

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

if(val==stack[top])

return 0;

return 1;

}

char dfs(int i){

int k;

char m;

if(top==-1)

push(b[i]);

m=pop();

top--;

arr[p]=m;

p++;

for(int j=0;j<n;j++){

if(ajMat[i][j]=='y'){

if(unVisit(b[j])){

push(b[j]);

}

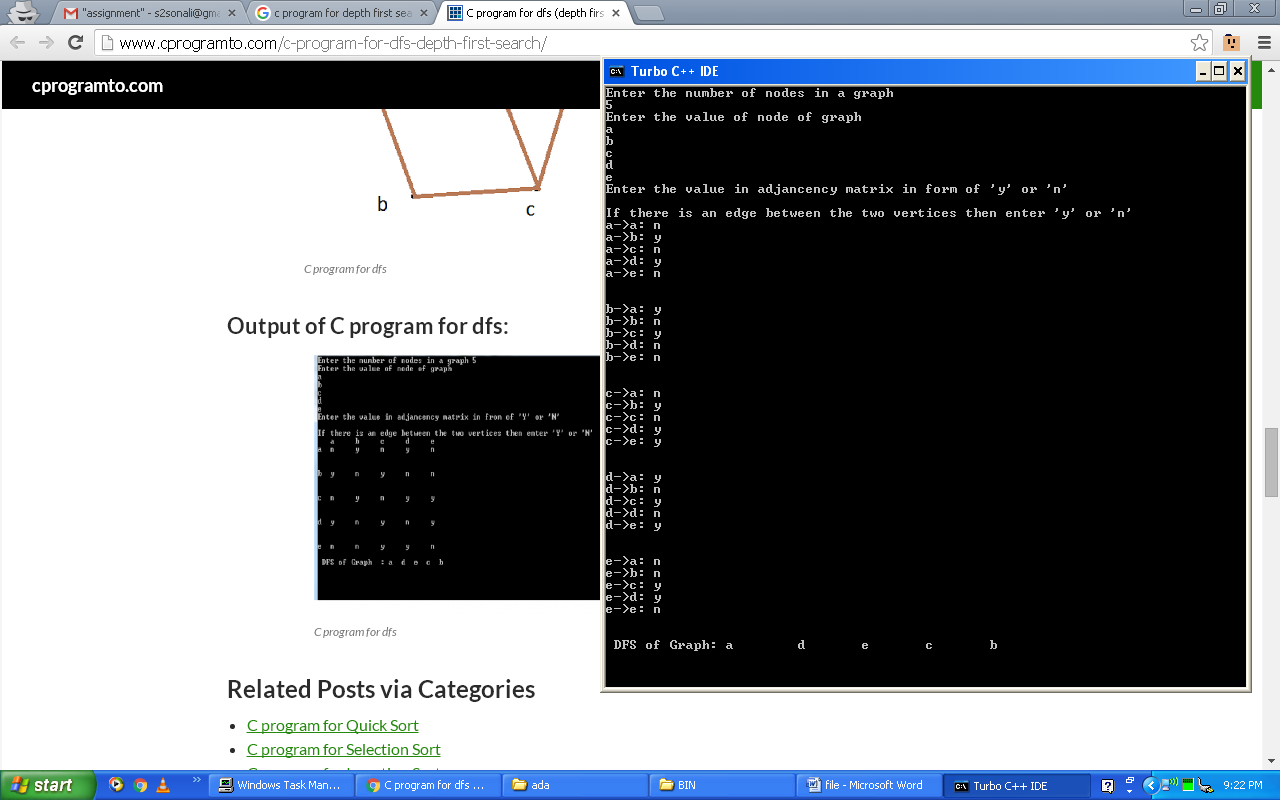
}

}

return stack[top];

}

**Output:**

****

**Q. Write a program to implement Topological Sort.**

**Code:**

#include<iostream.h>

#include<conio.h>

#define MAX\_NODE 50

struct node{

int vertex;

node \*next;

};

node \*adj[MAX\_NODE];

node \*sngllist=NULL;

int totNodes;

const int ready=1,wait=2,processed=3;

int status[MAX\_NODE];

void createGraph(){

node \*new1,\*last;

int neighbor,neighbor\_val;

cout<<"\n\n---Graph Creation--\n\n"; cout<<"Enter total nodes in graph: ";

cin>>totNodes;

for(int i=1;i<=totNodes;i++){

last=NULL;

cout<<"\nEnter no. of nodes in adjacency list of node "<<i<<"\n";

cout<<"-->That is Total Neighbours of "<<i<<": ";

cin>>neighbor;

for(int j=1;j<=neighbor;j++){

cout<<"Enter neighbor #"<<j<<": ";

cin>>neighbor\_val;

new1=new node; new1->vertex=neighbor\_val;

new1->next=NULL;

if(adj[i]==NULL)

adj[i]=last=new1;

else{

last->next=new1;

last=new1;

}

}

}

}

void dfsVisit(int i){

node \*tmp,\*new1;

int v; tmp=adj[i];

while(tmp!=NULL){

v=tmp->vertex;

if(status[v]==ready)

dfsVisit(v);

tmp=tmp->next;

}

new1=new node; new1->vertex=i;

new1->next=sngllist; sngllist=new1;

status[i]=processed;

}

void topologicalSort(){

int i;

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

status[i]=ready;

for(i=1;i<=totNodes;i++){

if(status[i]==ready)

dfsVisit(i);

}

while(sngllist!=NULL){

cout<<" "<<sngllist->vertex; sngllist=sngllist->next; }

}

void main(){

clrscr();

cout<<"\*\*\*\*\*Topological Sort\*\*\*\*\*\n";

createGraph();

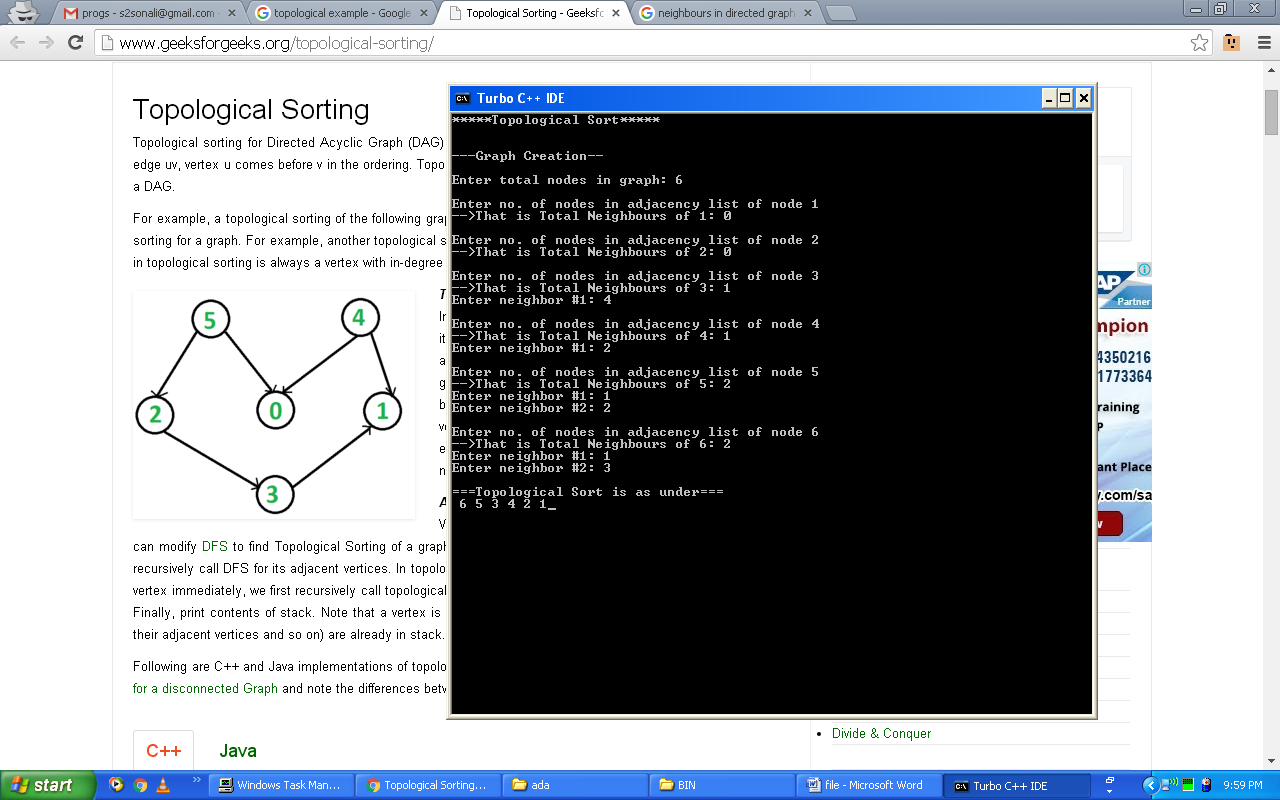
cout<<"\n===Topological Sort is as under===\n";

topologicalSort();

getch();

}

**Output:**

****

**Q. Write a program to implement Rod Cutting problem.**

**Code:**

#include<iostream.h>

#include<string.h>

#include<math.h>

#include<conio.h>

#include<limits.h>

int max(int a,int b){

return (a>b)?a:b;

}

int cutRod(int price[],int n){

if(n<=0)

return 0;

int max\_val=INT\_MIN;

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

max\_val=max(max\_val,price[i]+cutRod(price,n-i-1));

return max\_val;

}

void main(){

clrscr();

int arr[]={1,5,8,9,10,17,17,20};

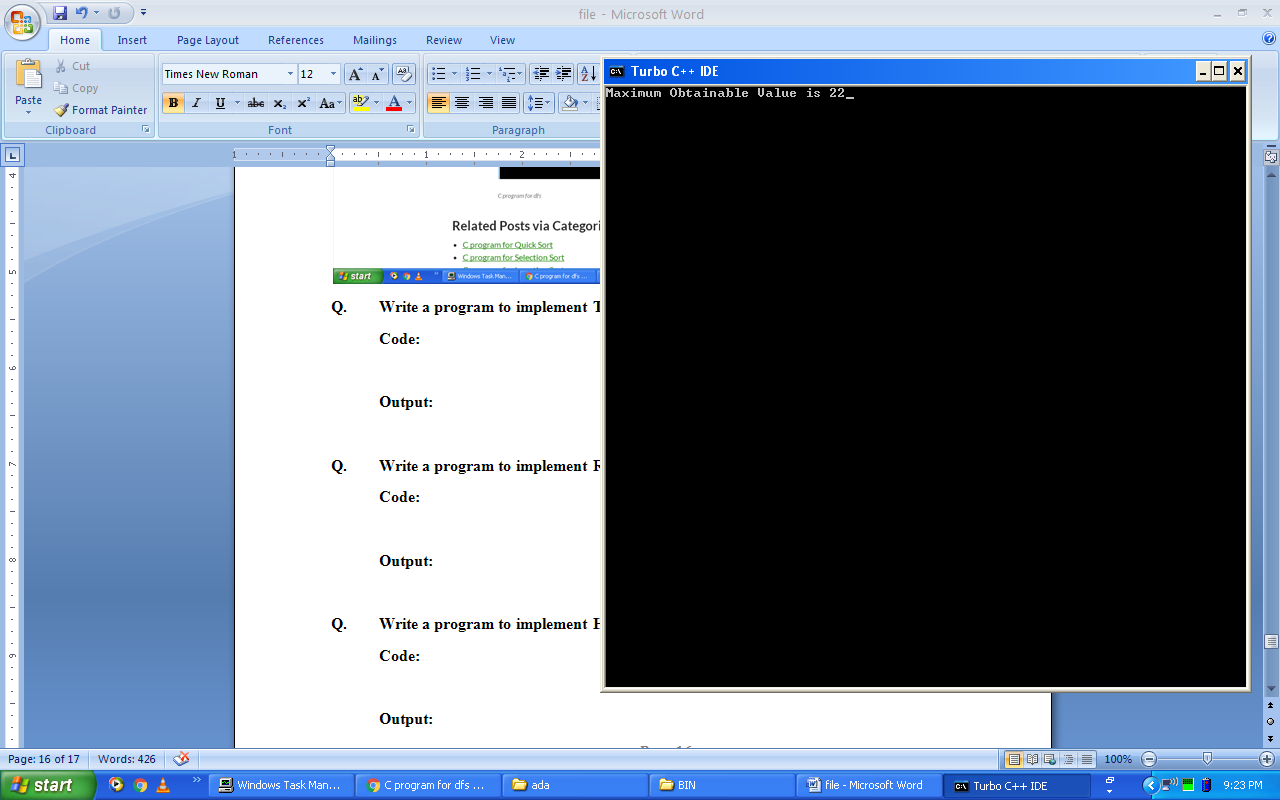
int size=sizeof(arr)/sizeof(arr[0]);

cout<<"Maximum Obtainable Value is "<<cutRod(arr,size);

getch();

}

**Output:**

****

**Q. Write a program to implement Huffman Code algorithm.**

**Code:**

#include<iostream.h>

#include<conio.h>

#include<malloc.h>

int k,f,str[20];

int i=0;

char ch;

struct NODE{

char data;

int freq;

struct NODE \*link,\*right,\*left;

};

typedef NODE node;

node \*start, \*last, \*s, \*a, \*p, \*q, \*z, \*t, \*min, \*min1;

node \*getnode(){

node \*p;

p=(node\*)malloc(sizeof(node));

return p;

}

void queue(){

if(start==NULL){

p=getnode();

cout<<"\nEnter the character: ";

cin>>ch;

p->data=ch;

cout<<"Enter its frequency: ";

cin>>f;

p->freq=f;

p->link=NULL;

p->right=NULL;

p->left=NULL;

start=p;

}

else{

s=start;

q=getnode();

cout<<"\nEnter the character : ";

cin>>ch;

q->data=ch;

cout<<"Enter frequency : ";

cin>>f;

q->freq=f;

q->link=NULL;

q->right=NULL;

q->left=NULL;

if(start->freq>q->freq){

q->link=start;

start=q;

}

else{

while((s->freq<q->freq) && (s!=NULL)){

a=s;

s=s->link;

}

q->link=a->link;

a->link=q;

}

}

}

void traverse(){

node \*ptr;

ptr=start;

while(ptr!=NULL)

ptr=ptr->link;

}

node\* extract\_min(){

t=start;

start=start->link;

t->link=NULL;

return t;

}

void insert(node \*z){

if(start==NULL)

start=z;

else{

s=start;

if(start->freq>z->freq){

z->link=start;

start=z;

}

else{

while((s->freq<z->freq) && (s!=NULL)){

a=s;

s=s->link;

}

z->link=a->link;

a->link=z;

}

}

}

node\* huffman(){

for(int y=0; y<(k-1);y++){

z=getnode();

min=extract\_min();

z->left=min;

min1=extract\_min();

z->right=min1;

z->freq=(min->freq)+(min1->freq);

z->data='A';

insert(z);

traverse();

}

return extract\_min();

}

void preorder(node\* t){

if(t!=NULL){

preorder(t->left);

preorder(t->right);

}

}

void print(node \*ptr,char x){

if(ptr->data!=x){

if(ptr->left){

str[i++]=0;

print(ptr->left,x);

}

if(ptr->right){

str[i++]=1;

print(ptr->right,x);

}

i--;

}

else{

cout<<"Huffman code for '"<<x<<"':";

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

cout<<str[p];

return;

}

}

void main(){

clrscr();

int i,j; k=0;

char z; char ch='y';

start=NULL;

cout<<"\nHow many characters do you wish to enter :";

cin>>i;

k=k+i;

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

queue();

traverse(); node \*l;

l=huffman(); traverse();

preorder(l); ch='y';

cout<<"\nHuffman code";

do{

cout<<"\nEnter the character: ";

cin>>z; print(l,z);

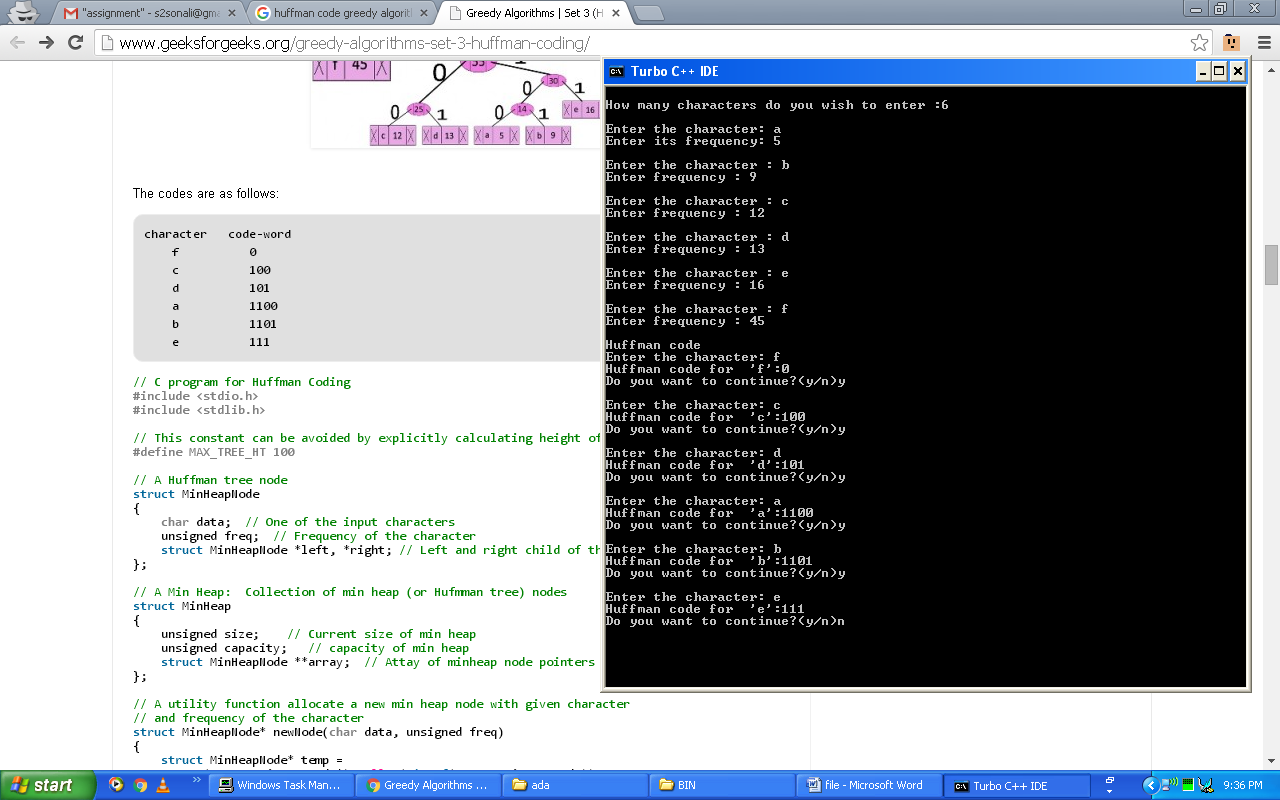
cout<<"\nDo you want to continue?(y/n)"; cin>>ch;

}while(ch=='y' || ch=='Y');

getch();

}

**Output:**

****