**PRACTICAL -1**

**Aim:** Write a C/C++ program to implement Bubble, Selection sort and measure time complexity.

**Code:**

/\* Bubble Sort.....!!!!\*/

#include <stdio.h>

#include <conio.h>

// ------------ MAIN SECTION -------------

void main()

{

clrscr();

int n,i,temp,a[20],j,k;

printf("Enter No. Of Elements:-->");

scanf("%d",&n);

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

{

printf("Enter A[%d]:-->",i);

scanf("%d",&a[i]);

}

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

{

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

{

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

{

temp=a[j];

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

a[j+1]=temp;

}

}

printf("\nPASS :--> %d :: ",i+1);

for(k=0;k<n;k++)printf("%d ",a[k]);

}

getch();

}

**Output:**

Output:

Enter No. Of Elements:-->5

Enter A[0]:-->2

Enter A[1]:-->6

Enter A[2]:-->7

Enter A[3]:-->3

Enter A[4]:-->5

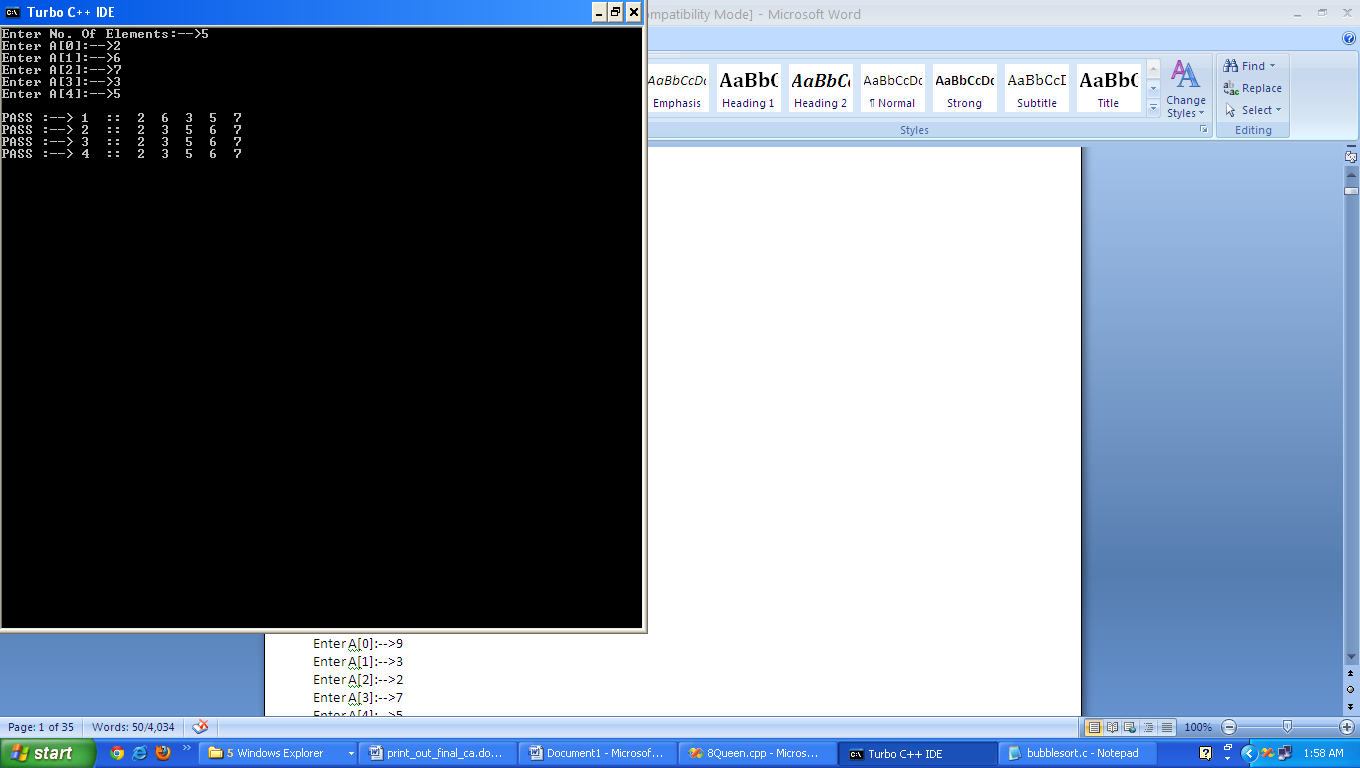
PASS :--> 1 :: 2 6 3 5 7

PASS :--> 2 :: 2 3 5 6 7

PASS :--> 3 :: 2 3 5 6 7

PASS :--> 4 :: 2 3 5 6 7

**Screen shot:**



**Code:**

/\* Selection Sort.....!!!!!\*/

#include <stdio.h>

#include <conio.h>

// ------------ MAIN SECTION -------------

void main()

{

clrscr();

int n,i,temp,a[20],j,k,min;

printf("Enter No. Of Elements:-->");

scanf("%d",&n);

printf("\n");

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

{

printf("Enter A[%d]:-->",i);

scanf("%d",&a[i]);

}

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

{

min=i;

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

{

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

}

if(a[i]!=a[min])

{

temp=a[i];

a[i]=a[min];

a[min]=temp;

}

printf("\nPASS :--> %d :: ",i+1);

for(k=0;k<n;k++) printf("%d ",a[k]);

}

getch();

}

**Output:**

Output:

Enter No. Of Elements:-->5

Enter A[0]:-->5

Enter A[1]:-->4

Enter A[2]:-->8

Enter A[3]:-->6

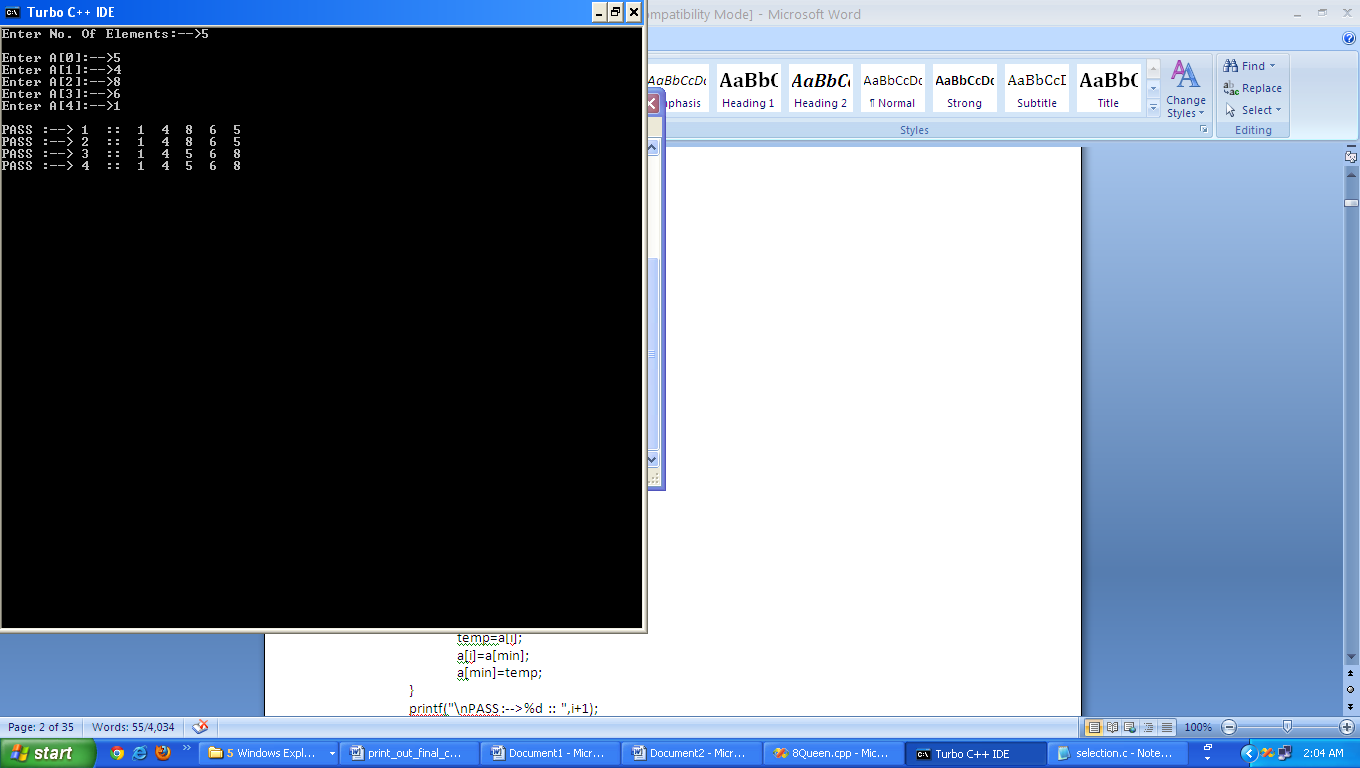
Enter A[4]:-->1

PASS :--> 1 :: 1 4 8 6 5

PASS :--> 2 :: 1 4 8 6 5

PASS :--> 3 :: 1 4 6 6 8

PASS :--> 4 :: 1 4 6 6 8

**Screen shot:**

**PRACTICAL -2**

**Aim:** Write a C/C++ program to implement the Quick Sort and measure time complexity.

**Code:**

/\* Program of sorting using quick sort through recursion \*/

#include<stdio.h>

#include<conio.h>

#define MAX 30

enum bool { FALSE,TRUE };

void display(int arr[],int low,int up)

{

int i;

for(i=low;i<=up;i++)

printf("%d ",arr[i]);

}

void quick(int arr[],int low,int up)

{

int piv,temp,left,right;

enum bool pivot\_placed=FALSE;

left=low;

right=up;

piv=low; /\*Take the first element of sublist as piv \*/

if(low>=up)

return;

printf("\nSublist : ");

display(arr,low,up);

/\*Loop till pivot is placed at proper place in the sublist\*/

while(pivot\_placed==FALSE)

{

/\*Compare from right to left \*/

while( arr[piv]<=arr[right] && piv!=right )

right=right-1;

if( piv==right )

pivot\_placed=TRUE;

if( arr[piv] > arr[right] )

{

temp=arr[piv];

arr[piv]=arr[right];

arr[right]=temp;

piv=right;

}

/\*Compare from left to right \*/

while( arr[piv]>=arr[left] && left!=piv )

left=left+1;

if(piv==left)

pivot\_placed=TRUE;

if( arr[piv] < arr[left] )

{

temp=arr[piv];

arr[piv]=arr[left];

arr[left]=temp;

piv=left;

}

}/\*End of while \*/

printf(" -> Pivot Placed is %d -> ",arr[piv]);

display(arr,low,up);

printf("\n");

quick(arr,low,piv-1);

quick(arr,piv+1,up);

}/\*End of quick()\*/

void main()

{

int array[MAX],n,i;

clrscr();

printf("Enter the number of elements : ");

scanf("%d",&n);

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

{

printf("Enter element %d : ",i+1);

scanf("%d",&array[i]);

}

printf("Unsorted list is :\n");

display(array,0,n-1);

printf("\n");

quick(array,0,n-1);

printf("Sorted list is :\n");

display(array,0,n-1);

printf("\n");

getch();

}

**Output:**

Enter element 1 : 5

Enter element 2 : 4

Enter the number of elements : 8

Enter element 3 : 6

Enter element 4 : 3

Enter element 5 : 1

Unsorted list is :

4 8 6 3 1

Sublist : 4 8 6 3 1 -> Pivot Placed is 4 -> 1 3 4 6 8

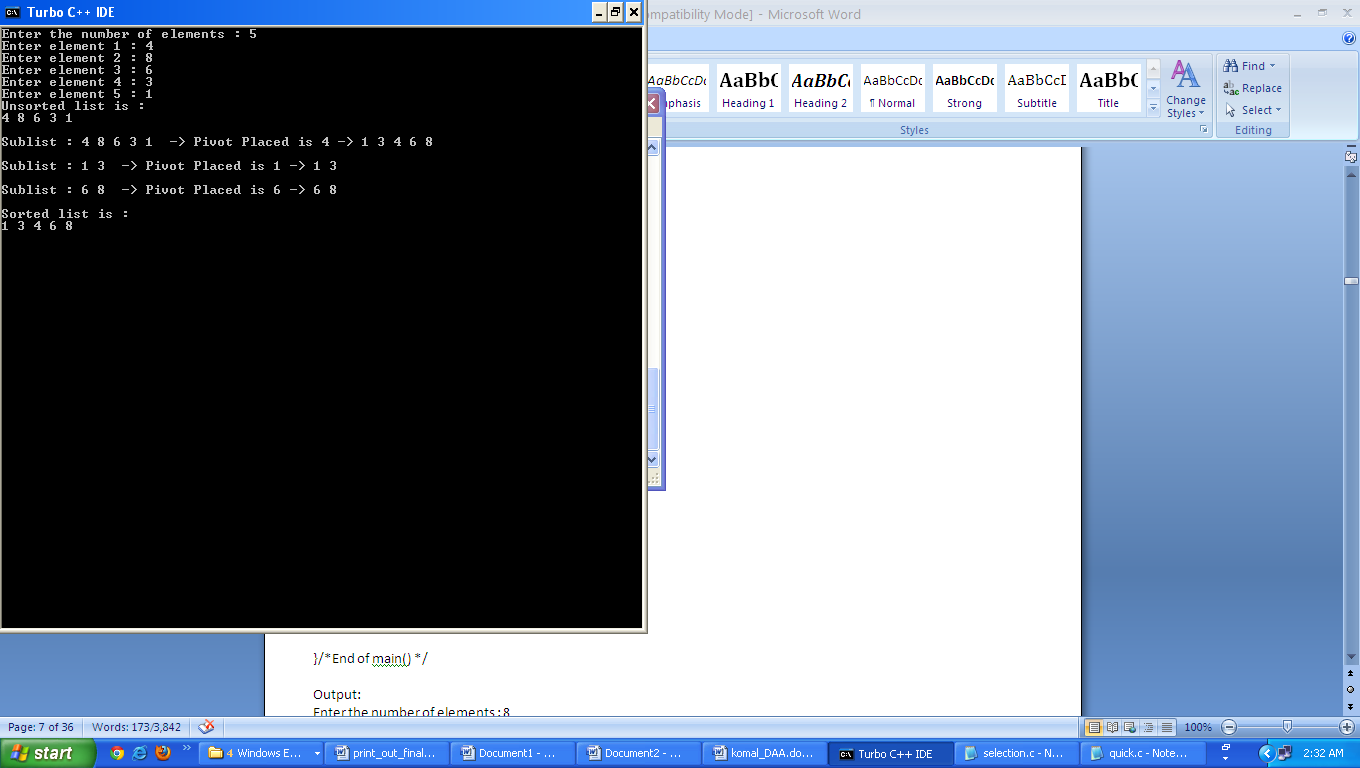
Sublist : 1 3-> Pivot Placed is 1 -> 1 3

Sublist : 6 8-> Pivot Placed is 6 -> 1 2 3

Sorted list is :

1 3 4 6 8

**Screen shot:**

****

**PRACTICAL -3**

**Aim:** Write a C/C++ program to implement the Merge Sort using Divide and conquer strategy.

**Code:**

#include<stdio.h>

#include<conio.h>

void combine(int a[10],int low, int mid, int high)

{

int i,j,k;

int temp[10];

k=low;

i=low;

j=mid+1;

while(i<=mid && j<=high)

{

if(a[i]<=a[j])

{

temp[k]=a[i];

i++;

k++;

}

else

{

temp[k]=a[j];

j++;

k++;

}

}

while(i<=mid)

{

temp[k]=a[i];

i++;

k++;

}

while(j<=high)

{

temp[k]=a[j];

j++;

k++;

}

//copy the element from temp to A

for(k=low;k<=high;k++)

{

a[k]=temp[k];

}

}

void mergesort(int a[10],int low, int high)

{

int mid;

if(low<high)

{

mid=(low+high)/2;

mergesort(a,low,mid);

mergesort(a,mid+1,high);

combine(a,low,mid,high);

}

}

void main()

{

int i,j,mid,high,low=0,k,a[10],n;

clrscr();

printf("Enter Array size");

scanf("%d",&n);

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

{

printf("enter array element");

scanf("%d",&a[i]);

}

high=n-1;

mergesort(a,low,high);

//Display sorted array

printf("SORTED ARRAY:");

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

{

printf("a[%d]=%d\n",i,a[i]);

}

getch();

}

**Output:**

Enter Array size:5

Enter array element:9

Enter array element:6

Enter array element:2

Enter array element:0

Enter array element:3

SORTED ARRAY:

a[0]=0

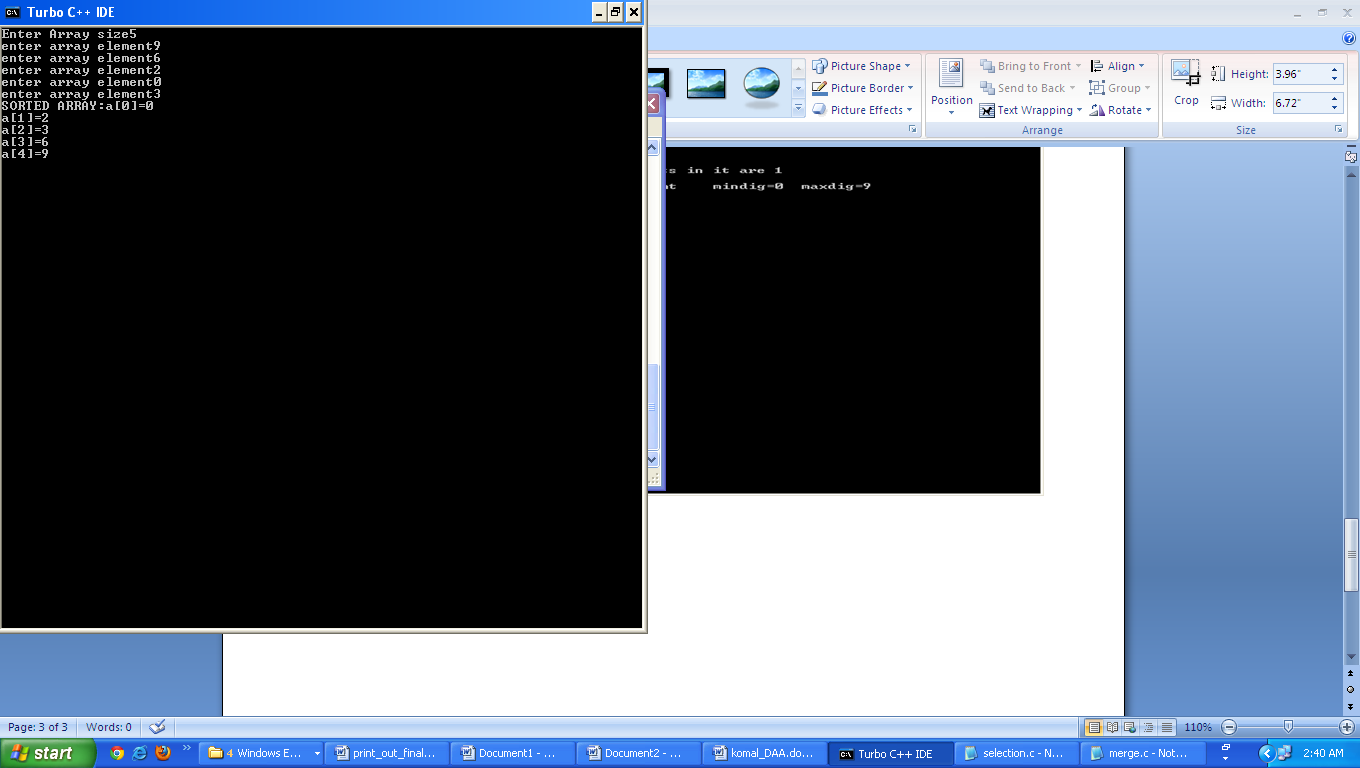
a[1]=2

a[2]=3

a[3]=6

a[4]=9

**Screen shot:**

****

**PRACTICAL -4**

**Aim:** Write a program to implement Binary Search using C or C++ programming.

**Code:**

/\* Program for binary search\*/

#include <stdio.h>

#include <conio.h>

void main()

{

int arr[20],start,end,middle,n,i,item;

clrscr();

printf("How many elements you want to enter in the array : ");

scanf("%d",&n);

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

{

printf("Enter element %d : ",i+1);

scanf("%d",&arr[i]);

}

printf("Enter the element to be searched : ");

scanf("%d",&item);

start=0;

end=n-1;

middle=(start+end)/2;

while(item != arr[middle] && start <= end)

{

if(item > arr[middle])

start=middle+1;

else

end=middle-1;

middle=(start+end)/2;

}

if(item==arr[middle])

printf("%d found at position %d\n",item,middle+1);

if(start>end)

printf("%d not found in array\n",item);

getch();

}

**Output:**

How many elements you want to enter in the array : 5

Enter element 1 : 1

Enter element 2 : 2

Enter element 3 : 3

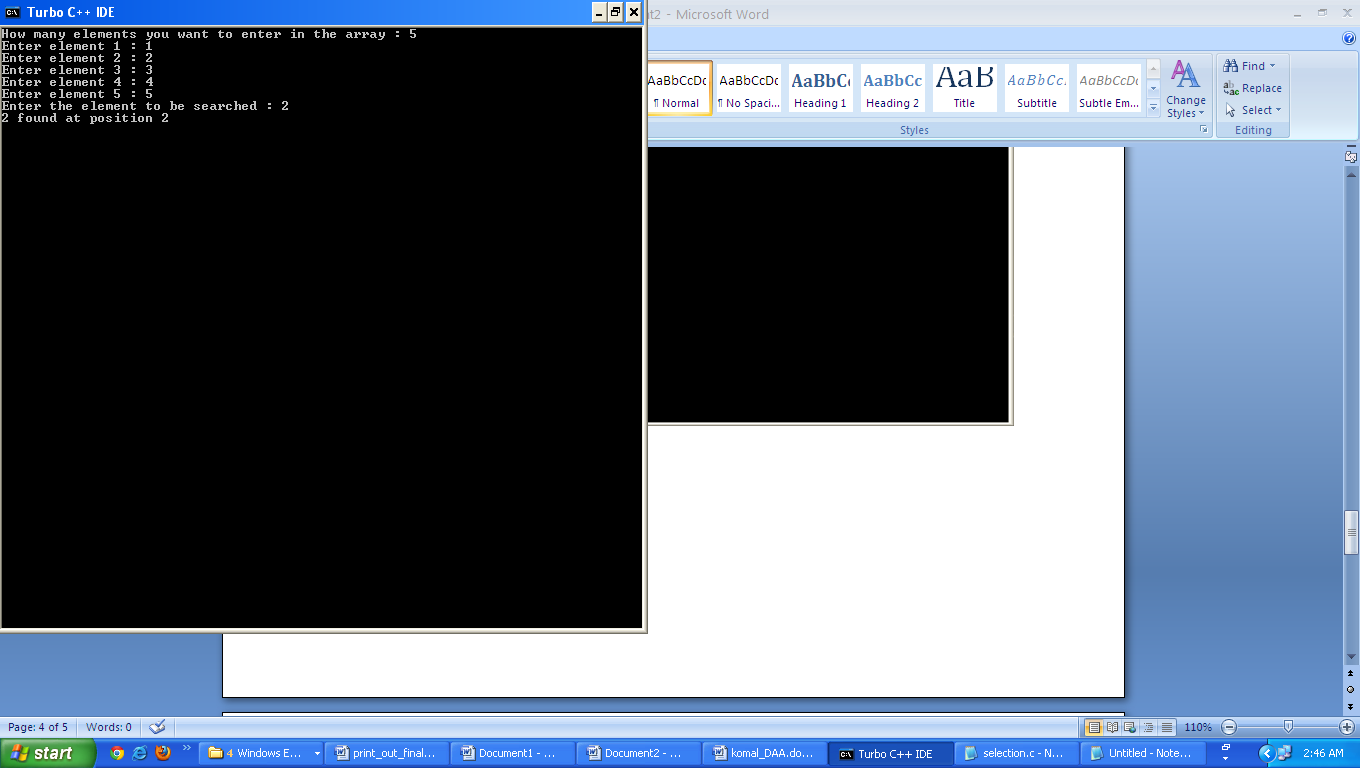
Enter element 4 : 4

Enter element 5 :5

Enter the element to be searched : 2

2 found at position 2

**Screen shot:**

****

**PRACTICAL -5**

**Aim:** Write a C/C++ program to implement Kruskal’s algorithm using Greedy strategy.

**Code:**

/\*Implementation of Kruskal's Algorithm\*/

#include<stdio.h>

#define INFINITY 999

typedef struct Graph

{

int v1;

int v2;

int cost;

}GR;

GR G[20];

int tot\_edges,tot\_nodes;

void create();

void spanning\_tree();

int Minimum(int);

void main()

{

printf("\n\t Graph Creation by adjacency matrix ");

create();

spanning\_tree();

}

void create()

{

int k;

printf("\n Enter Total number of nodes: ");

scanf("%d",&tot\_nodes);

printf("\n Enter Total number of edges: ");

scanf("%d",&tot\_edges);

for(k=0;k<tot\_edges;k++)

{

printf("\n Enter Edge in (V1 V2)form ");

scanf("%d%d",&G[k].v1,&G[k].v2);

printf("\n Enter Corresponding Cost ");

scanf("%d",&G[k].cost);

}

}

void spanning\_tree()

{

int count,k,v1,v2,i,j,tree[10][10],pos,parent[10];

int sum;

int Find(int v2,int parent[]);

void Union(int i,int j,int parent[]);

count=0;

k=0;

sum=0;

for(i=0;i<tot\_nodes;i++)

parent[i]=i;

while(count!=tot\_nodes-1)

{

pos=Minimum(tot\_edges);//finding the minimum cost edge

if(pos==-1)//Perhaps no node in the graph

break;

v1=G[pos].v1;

v2=G[pos].v2;

i=Find(v1,parent);

j=Find(v2,parent);

if(i!=j)

{

tree[k][0]=v1;//storing the minimum edge in array tree[]

tree[k][1]=v2;

k++;

count++;

sum+=G[pos].cost;//accumulating the total cost of MST

Union(i,j,parent);

}

G[pos].cost=INFINITY;

}

if(count==tot\_nodes-1)

{

printf("\n Spanning tree is...");

printf("\n--------------------------\n");

for(i=0;i<tot\_nodes-1;i++)

{

printf("[%d",tree[i][0]);

printf(" - ");

printf("%d",tree[i][1]);

printf("]");

}

printf("\n--------------------------");

printf("\nCost of Spanning Tree is = %d",sum);

}

else

{

printf("There is no Spanning Tree");

}

}

int Minimum(int n)

{

int i,small,pos;

small=INFINITY;

pos=-1;

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

{

if(G[i].cost<small)

{

small=G[i].cost;

pos=i;

}

}

return pos;

}

int Find(int v2,int parent[])

{

while(parent[v2]!=v2)

{

v2=parent[v2];

}

return v2;

}

void Union(int i,int j,int parent[])

{

if(i<j)

parent[j]=i;

else

parent[i]=j;

}

**Output:**

Enter total no. of nodes : 1

Enter total no.of edges : 3

Enter Corresponding cost 6

Enter edge in (V1 V2) form 4

Enter corresponding cost 4

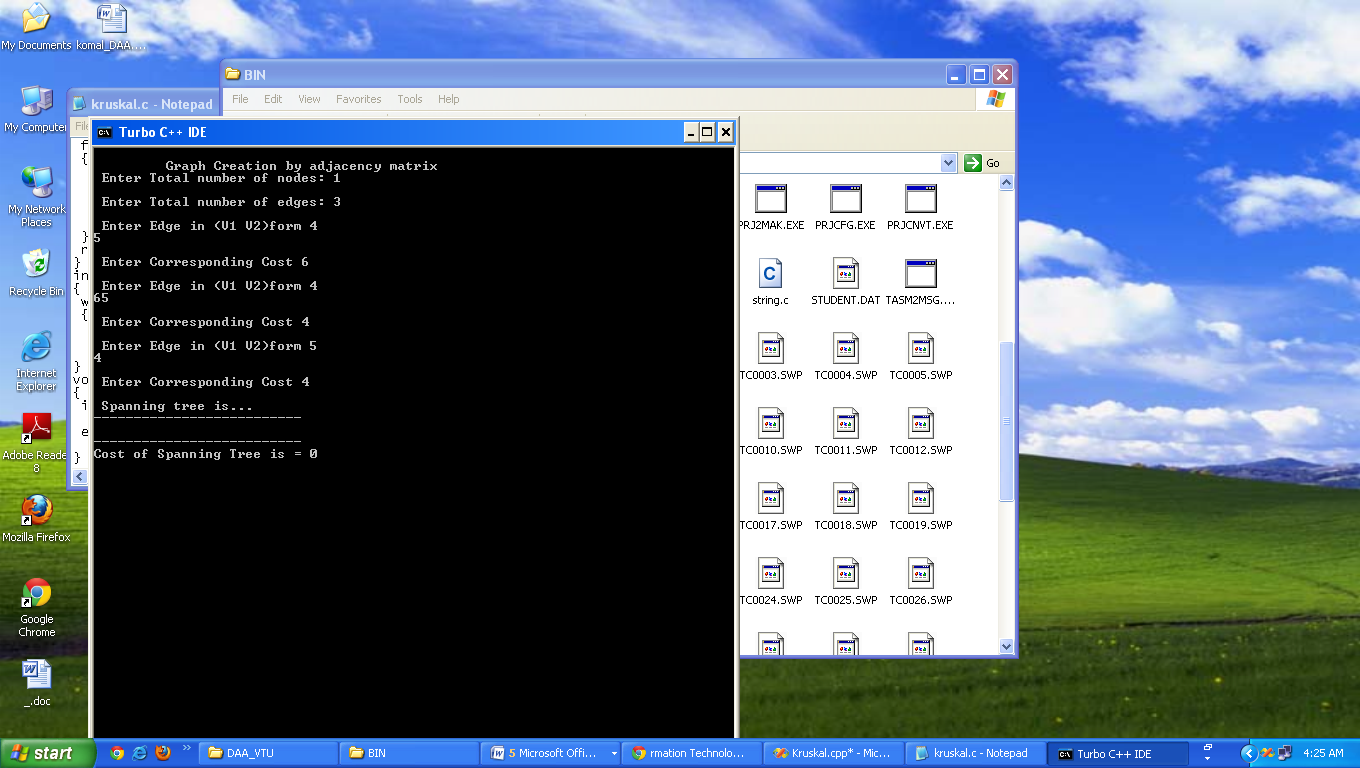
Enter edge in (V1 V2) form 5

Enter corresponding cost 4

Spanning tree is…

Cost of spanning tree is = 0

**Screen shot:**

****

**PRACTICAL -6**

**Aim:** Write a C/C++ program to implement Prim’s algorithm using Greedy strategy.

**Code:**

/\*This program is to implement Prim's Algorithm \*/

# include<stdio.h>

# include<conio.h>

# define SIZE 20

# define INFINITY 32767

/\* This function finds the minimal spanning tree by Prim's Algorithm \*/

void Prim(int G[][SIZE], int nodes)

{

int select[SIZE], i, j, k;

int min\_dist, v1, v2,total=0;

for (i=0 ; i<nodes ; i++) // Initialize the selected vertices list

select[i] = 0;

printf("\n\n The Minimal Spanning Tree Is :\n");

select[0] = 1;

for (k=1 ; k<nodes ; k++)

{

min\_dist = INFINITY;

for (i=0 ; i<nodes ; i++) // Select an edge such that one vertex is

{ // selected and other is not and the edge

for (j=0 ; j<nodes ; j++) // has the least weight.

{

if (G[i][j] && ((select[i] && !select[j]) || (!select[i] && select[j])))

{

if (G[i][j] < min\_dist)//obtained edge with minimum wt

{

min\_dist = G[i][j];

v1 = i;

v2 = j; //picking up those vertices

}

}

}

}

printf("\n Edge (%d %d )and weight = %d",v1,v2,min\_dist);

select[v1] = select[v2] = 1;

total =total+min\_dist;

}

printf("\n\n\t Total Path Length Is = %d",total);

}

void main()

{

int G[SIZE][SIZE], nodes;

int v1, v2, length, i, j, n;

clrscr();

printf("\n\t Prim'S Algorithm\n");

nodes=4;

n=4;

for (i=0 ; i<nodes ; i++) // Initialize the graph

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

G[i][j] = 0;

G[0][1]=G[1][0]=1;

G[0][2]=G[2][0]=5;

G[0][3]=G[3][0]=2;

G[2][3]=G[3][2]=3;

printf("\n Graph is created");

getch();

printf("\n\t");

Prim(G,nodes);

getch();

}

**Output:**

graph is created

The minimal spanning tree is :

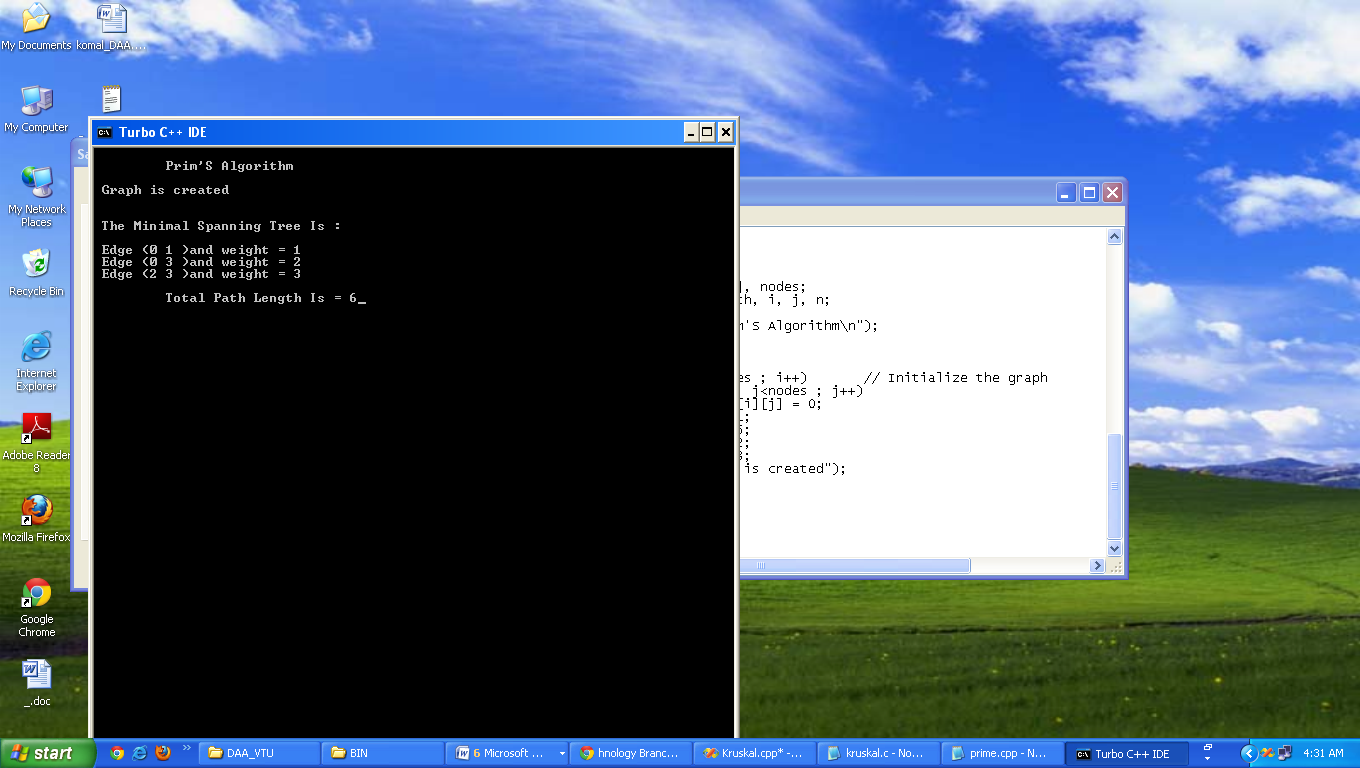
Edge<0 1> and weight = 1

Edge<0 3> and weight = 2

Edge<2 3> and weight = 3

Total path length = 6

**Screen shot:**

****

**PRACTICAL -7**

**Aim:** Write a C/C++ program to implement knapsack problem using Dynamic Programming.

**Code:**

/\*KNAPSACK.....!!!!\*/

#include<stdio.h>

#include<conio.h>

#define P printf("\n")

// --------- DECLARATION SECTION --------------

int w[12],v[12],val[12][12],W,n,i,j,k;

int max(int,int);

void knapsack();

void disp();

// -------------- MAIN SECTION ---------------

void main()

{

clrscr();

printf("Enter number of Objects(wi and vi):-->");

scanf("%d",&n);

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

{

printf("Enter w%d :--> ",i);

scanf("%d",&w[i]);

printf("Enter v%d :--> ",i);

scanf("%d",&v[i]);

}

printf("Enter W:-->");

scanf("%d",&W);

knapsack();

getch();

}

// ------------- Apply Dynamic Programming --------------

void knapsack()

{

int a1,a2,x,y;

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

{

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

{

if(j==0)

{

val[i][j]=0;

}

if(j!=0 && i==1)

{

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

}

if(j!=0 && i!=1)

{

a1=val[i-1][j]; // Memory Function

a2=(val[i-1][j-w[i]])+v[i];

if(j-w[i] < 0)

{

val[i][j]=a1;

}

else

{

val[i][j]=max(a1,a2);

}

}

}

}

disp();

}

int max(int m,int n)

{

if(m>=n)

return m;

else

return n;

}

void disp()

{

P;P;

printf(" ");

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

printf("%02d ",i);

P;

printf(" ");

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

printf("----");

P;

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

{

printf("%02d %02d | ",w[i],v[i]);

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

{

printf("%02d ",val[i][j]);

}

printf("\n");

}

}

**Output:**

number of Enter objects<wi and vi>:-- >5

Enter w1 :-- >1

Enter v1 :-- >2

Enter w2 :-- 2

Enter v2 :-- >3

Enter w3 :-- >3

Enter v3 :-- >4

Enter w4 :-- >4

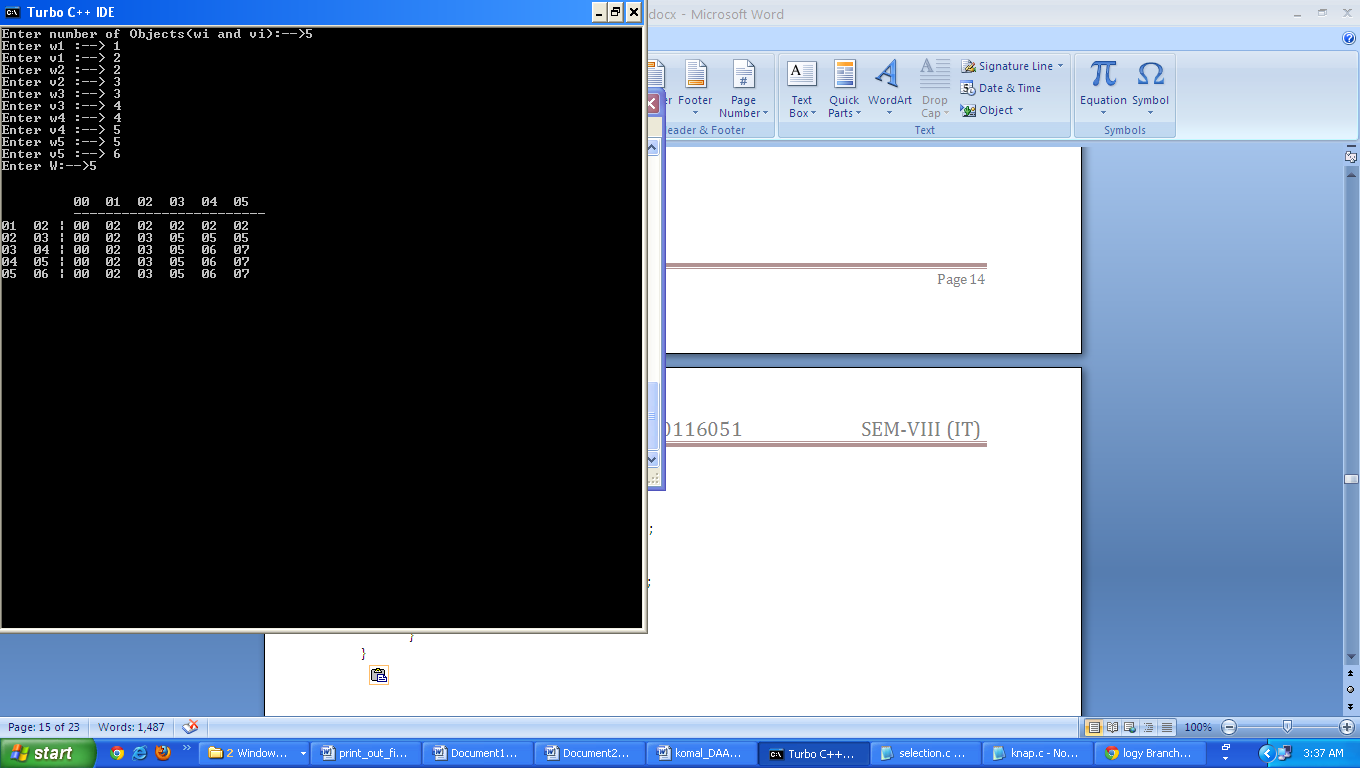
Enter v4 :-- >5

Enter w5 :-- >5

Enter v5 :-- >6

Enter w :-- >5

**Screen shot:**

****

**PRACTICAL -8**

**Aim:** Write a C/C++ program to implement Matrix Chain Multiplication using Dynamic Programming.

**Code:**

/\* MATRIX MULTIPLICATION....!!!\*/

#include <stdio.h>

#include <conio.h>

#include <string.h>

#define P printf("\n")

// ------------- DECLARATION SECTION -----------

int n,i,j,k,l,s,round;

int d[20],m[20][20],mul[5];

void PrintLine();

// ------------ MAIN SECTION -------------

void main()

{

clrscr();

printf("Enter Number of Matrix To Multiply :--> ");

scanf("%d",&n);

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

{

printf("\nEnter D%d :--> ",i);

scanf("%d",&d[i]);

}

round=n-1;

s=0;

PrintLine();

while(s<=round)

{

if(s==0)

{

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

{

m[i][i]=0;

printf("\nm%d%d=%d",i,i,m[i][i]);

}

PrintLine();

getch();

}

// ---------------------------------------------

if(s==1)

{

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

{

m[i][i+1]=d[i-1]\*d[i]\*d[i+1];

printf("\nm%d%d=%d",i,i+1,m[i][i+1]);

}

PrintLine();

getch();

getch();

}

// ---------------------------------------------

if(s>1)

{

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

{

l=i+1;

j=0;

for(k=i;k<(s+i);k++)

{

mul[j]=(m[i][k]+m[l][s+i]+

d[i-1]\*d[k]\*d[s+i]);

l++;

j++;

}

int temp,x,y;

for(x=0;x<j-1;x++)

{

for(y=x+1;y<j;y++)

{

if(mul[x]>mul[y])

{

temp=mul[x];

mul[x]=mul[y];

mul[y]=temp;

}

}

}

m[i][i+s]=mul[0];

printf("\nm%d%d=%d",i,i+s,m[i][i+s]);

}

PrintLine();

getch();

}

s++;

}

printf("\n Minimum Number Of Multiplications :--> %d ",m[1][n]);

getch();

}

void PrintLine()

{

printf("\n-----------");

}

**Output:**

Enter Number of Matrix To Multiply :--> 3

Enter D0 :--> 2

Enter D1 :--> 2

Enter D2 :--> 2

Enter D3 :--> 2

-----------

m11=0

m22=0

m33=0

-----------

m12=8

m23=8

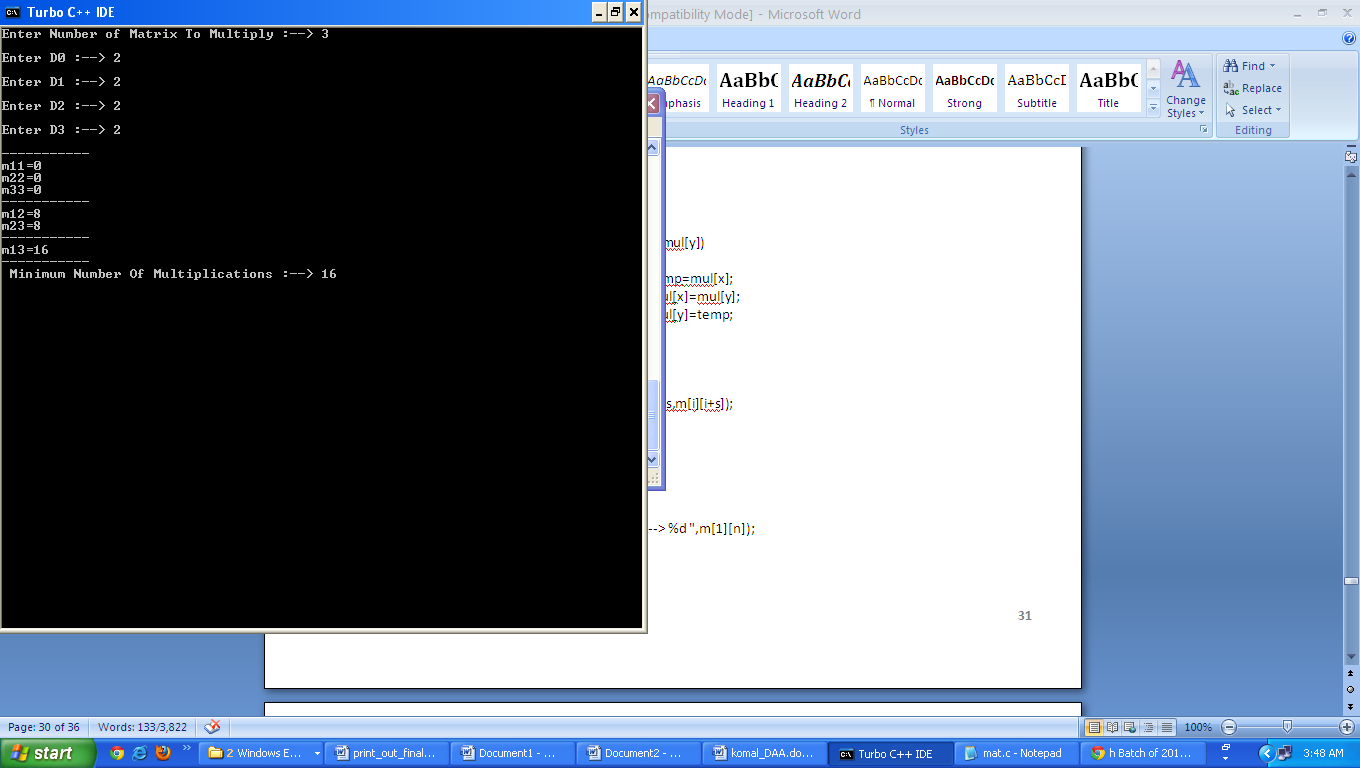
-----------

m13=16

-----------

Minimum Number Of Multiplications :--> 16

**Screen shot:**

****

**PRACTICAL -9**

**Aim:** Write a C/C++ program to implement Depth First Search and Breath First Search Algorithm.

**Code:**

#include<stdio.h>

int q[ 20 ], top = -1, front = -1, rear = -1, a[ 20 ][ 20 ], vis[ 20 ], stack[ 20 ];

int delete();

void add ( int item );

void bfs( int s, int n );

void dfs( int s, int n );

void push( int item );

int pop();

void main()

{

int n, i, s, ch, j;

char c, dummy;

clrscr();

printf( "ENTER THE NUMBER VERTICES " );

scanf( "%d", &n );

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

{

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

{

printf( "(ENTER : IF Edge then 1 ELSE or No Edge 0) %d to %d = ", i, j );

scanf( "%d", &a[ i ][ j ] );

}

}

printf( "\nTHE ADJACENCY MATRIX IS\n" );

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

{

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

{

printf( " %d", a[ i ][ j ] );

}

printf( "\n" );

}

do

{

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

vis[ i ] = 0;

printf( "\nMENU" );

printf( "\n1.B.F.S" );

printf( "\n2.D.F.S" );

printf( "\nENTER YOUR CHOICE" );

scanf( "%d", &ch );

printf( "\nENTER THE SOURCE VERTEX :" );

scanf( "%d", &s );

switch ( ch )

{

case 1:

bfs( s, n );

break;

case 2:

dfs( s, n );

break;

}

printf( "\nDO U WANT TO CONTINUE(Y/N) ? " );

scanf( "%c", &dummy );

scanf( "%c", &c );

}

while ( ( c == 'y' ) || ( c == 'Y' ) );

getch();

}

void bfs( int s, int n )

{

int p, i;

add( s );

vis[ s ] = 1;

p = delete();

if ( p != 0 )

printf( " %d", p );

while ( p != 0 )

{

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

if ( ( a[ p ][ i ] != 0 ) && ( vis[ i ] == 0 ) )

{ add( i );

vis[ i ] = 1;

}

p = delete();

if ( p != 0 )

printf( " %d ", p );

}

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

if ( vis[ i ] == 0 )

bfs( i, n );

}

void add( int item )

{

if ( rear == 19 )

printf( "QUEUE FULL" );

else

{

if ( rear == -1 )

{

q[ ++rear ] = item;

front++;

}

else

q[ ++rear ] = item;

}

}

int delete()

{

int k;

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

return ( 0 );

else

{

k = q[ front++ ];

return ( k );

}

}

void dfs( int s, int n )

{

int i, k;

push( s );

vis[ s ] = 1;

k = pop();

if ( k != 0 )

printf( " %d ", k );

while ( k != 0 )

{

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

if ( ( a[ k ][ i ] != 0 ) && ( vis[ i ] == 0 ) )

{

push( i );

vis[ i ] = 1;

}

k = pop();

if ( k != 0 )

printf( " %d ", k );

}

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

if ( vis[ i ] == 0 )

dfs( i, n );

}

void push( int item )

{

if ( top == 19 )

printf( "Stack overflow " );

else

stack[ ++top ] = item;

}

int pop()

{

int k;

if ( top == -1 )

return ( 0 );

else

{

k = stack[ top-- ];

return ( k );

}

}

**Output:**

ENTER THE NUMBER VERTICES 4

(ENTER : IF Edge then 1 ELSE or No Edge 0) 1 to 1 = 0

(ENTER : IF Edge then 1 ELSE or No Edge 0) 1 to 2 = 1

(ENTER : IF Edge then 1 ELSE or No Edge 0) 1 to 3 = 1

(ENTER : IF Edge then 1 ELSE or No Edge 0) 1 to 4 = 0

(ENTER : IF Edge then 1 ELSE or No Edge 0) 1 to 5 = 0

(ENTER : IF Edge then 1 ELSE or No Edge 0) 2 to 1 = 1

(ENTER : IF Edge then 1 ELSE or No Edge 0) 2 to 2 = 0

(ENTER : IF Edge then 1 ELSE or No Edge 0) 2 to 3 = 1

(ENTER : IF Edge then 1 ELSE or No Edge 0) 2 to 4 = 0

(ENTER : IF Edge then 1 ELSE or No Edge 0) 2 to 5 = 0

(ENTER : IF Edge then 1 ELSE or No Edge 0) 3 to 1 = 1

(ENTER : IF Edge then 1 ELSE or No Edge 0) 3 to 2 = 1

(ENTER : IF Edge then 1 ELSE or No Edge 0) 3 to 3 = 0

(ENTER : IF Edge then 1 ELSE or No Edge 0) 3 to 4 = 0

(ENTER : IF Edge then 1 ELSE or No Edge 0) 3 to 5 = 1

(ENTER : IF Edge then 1 ELSE or No Edge 0) 4 to 1 = 0

(ENTER : IF Edge then 1 ELSE or No Edge 0) 4 to 2 = 0

(ENTER : IF Edge then 1 ELSE or No Edge 0) 4 to 3 = 0

(ENTER : IF Edge then 1 ELSE or No Edge 0) 4 to 4 = 0

(ENTER : IF Edge then 1 ELSE or No Edge 0) 4 to 5 = 1

THE ADJACENCY MATRIX IS

0 1 1 0 0

1 0 1 0 0

1 1 0 0 1

0 0 0 0 1

0 0 1 1 0

MENU

1.B.F.S

2.D.F.S

ENTER YOUR CHOICE1

ENTER THE SOURCE VERTEX :1

1 2 3 5 4

DO U WANT TO CONTINUE(Y/N) ? y

MENU

1.B.F.S

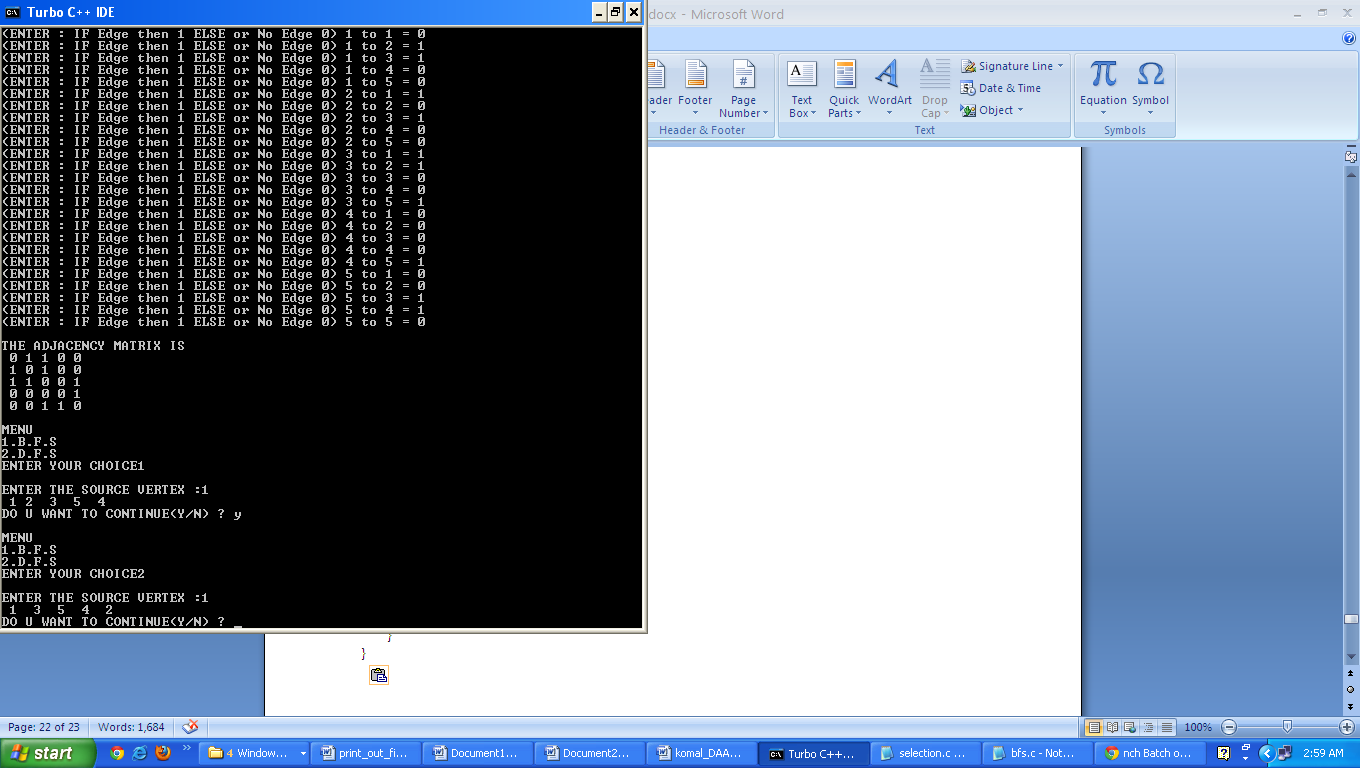
2.D.F.S

ENTER YOUR CHOICE2

ENTER THE SOURCE VERTEX :2

1 3 5 4 2

DO U WANT TO CONTINUE(Y/N) ? n

**Screen shot:**

**PRACTICAL -10**

**Aim:** Write a C/C++ program to implement Naive/ Rabin-Karp String Matching Algorithm.

**Code:**

/\* PROGRAM TO IMPLEMENT NAIVE STRING-MATCHING ALGORITHM....!!!! \*/

#include<conio.h>

#include<string.h>

#include<stdio.h>

void naive\_string\_matcher(char text[],char pat[])

{

char temp[100];

int n=strlen(text);

int m=strlen(pat);

int i,j,s,k;

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

{

for(j=s,k=0;j<m;j++,k++)

temp[k]=text[s+k];

temp[k]='\0';

if(strcmp(pat,temp)==0)

printf("\n PATTERN OCCURS WITH SHIFT :%d\n",s);

m++;

}

}

void main()

{

char text[100],pat[100];

clrscr();

printf("\n ENTER THE TEXT : ");

gets(text);

printf("\n ENTER THE PATTERN : ");

gets(pat);

naive\_string\_matcher(text,pat);

getch();

}

**Output:**

ENTER THE TEXT : komal

ENTER THE PATTERN : oma

PATTERN OCCURS WITH SHIFT :1

**Screen shot:**

