

## PROGRAM-1

11/11/20221

### AIM:

To write a menu driven program to perform operation on an array(insert,delete,display,sort,search).

### PROGRAM:

```
#include<stdio.h>
int a[50],i,j,n,k,pos,t;
void main()
{
void insert_beg();
void insert_mid();
void insert_end();
void delete_beg();
void delete_mid();
void delete_end();
void sort();
void search();
void display();
int ch;
    printf("Enter the size of the array: ");
    scanf("%d",&n);
    printf("\n Enter the elements to the array: ");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("\n MENU DRIVEN \n");
    printf("\n 1.To insert an element at the beginning.");
    printf("\n 2.To insert an element at middle.");
    printf("\n 3.To insert an element at the end");
    printf("\n 4.To delete an element at the beginning.");
    printf("\n 5.To delete an element at middle.");
    printf("\n 6.To delete an element at the end");
    printf("\n 7.To display the elements.");
    printf("\n 8.To sort an array.");
    printf("\n 9.To search an element.");
```

```

do
{

    printf("\n Enter your choice:");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1:insert_beg();
            break;
        case 2:insert_mid();
            break;
        case 3:insert_end();
            break;
        case 4:delete_beg();
            break;
        case 5:delete_mid();
            break;
        case 6:delete_end();
            break;
        case 7:display();
            break;
        case 8:sort();
            break;
        case 9:search();
            break;
        default:printf("\nInvalid choice.");
    }
}
while(ch<10);
}
void insert_beg()
{
    printf("\n Enter the element to be inserted at the beginning.");
    scanf("%d",&k);
    for(i=n;i>=0;i--)
    {
        a[i]=a[i-1];
    }
    a[0]=k;
    n=n+1;
}

```

```

}
void insert_mid()
{
    printf("\nEnter the element to be inserted at middle.");
    scanf("%d",&k);
    printf("\nEnter the position of element:");
    scanf("%d",&pos);
    for(i=n;i>=pos;i--)
    {
        a[i]=a[i-1];
    }
    a[pos-1]=k;
    n=n+1;
}
void insert_end()
{
    printf("\n Enter the element to be inserted:");
    scanf("%d",&k);
    a[n]=k;
    n=n+1;
}
void delete_beg()
{
    for(i=1;i<n;i++)
    {
        a[i-1]=a[i];
    }
    n=n-1;
    printf("Element deleted successfully");
}
void delete_mid()
{
    printf("Enter the position: ");
    scanf("%d",&pos);
    for(i=pos-1;i<n;i++)
    {
        a[i]=a[i+1];
    }
    printf("\nElement deleted successfully.");
    n=n-1;
}

```

```

}
void delete_end()
{
if(n==0)
{
printf("\nArray is empty");
}
else
{
n=n-1;
printf("\nElement deleted successfully");
}
}
void display()
{
int i;
if(n>0)
{
printf("\nArray elements are: ");
for(i=0; i<n; i++)
printf("%d\t ", a[i]);

}
else
{
printf("\nNo elements to display");
}
}
void sort()
{
for(i=0; i<n; i++)
{
for(int j=0; j<n-1; j++)
{
if(a[j]>a[j+1])
{
t=a[j];
a[j]=a[j+1];
a[j+1]=t;
}
}
}
}

```

```
    }  
}  
printf("\nElements sorted successfully");  
}  
void search()  
{  
    printf("\nArray Searching");  
    printf("\nEnter Element to be searched : ");  
    scanf("%d",&k);  
    printf("Elements Found at : ");  
    for(i=0;i<n;i++)  
    {        if(k==a[i])  
        printf("%d ",i+1);  
    }  
}
```

## OUTPUT:

```
Enter the size of the array: 5

Enter the elements to the array: 2 5 9 1 4

MENU DRIVEN

1.To insert an element at the beginning.
2.To insert an element at middle.
3.To insert an element at the end
4.To delete an element at the beginning.
5.To delete an element at middle.
6.To delete an element at the end
7.To display the elements.
8.To sort an array.
9.To search an element.
Enter your choice:1

Enter the element to be inserted at the beginning.11

Enter your choice:2

Enter the element to be inserted at middle.0

Enter the position of element:3

Enter your choice:3

Enter the element to be inserted:20
```

```
Enter the element to be inserted:20

Enter your choice:7

Array elements are: 11    2        0        5        9        1        4        20
Enter your choice:5
Enter the position: 5

Element deleted successfully.
Enter your choice:4
Element deleted successfully
Enter your choice:6

Element deleted successfully
Enter your choice:8

Elements sorted successfully
Enter your choice:9

Array Searching
Enter Element to be searched : 1
Elements Found at : 2
Enter your choice:7

Array elements are: 0    1        2        4        5
Enter your choice:Killed
```

## RESULT:

The program executed successfully and output is verified.

## PROGRAM-2

18/11/20221

### AIM:

write a program to merge two sorted arrays.

### PROGRAM:

```
#include <stdio.h>
void main()
{
    int a[25],b[25],c[50],i,j,n,m,k,l;
    printf("Enter the total number of elements in 1st array: ");
    scanf("%d",&n);
    printf("Enter the total number of elements in 2nd array: ");
    scanf("%d",&m);
    printf("Enter the elements of 1st sorted array: ");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("\n Enter the elements of 2nd sorted array: ");
    for(i=0;i<m;i++)
    {
        scanf("%d",&b[i]);
    }

    i=0,j=0,k=0;
    while(i<n && j<m)
    {
        if(a[i]>b[j])
        {
            c[k]=b[j];
            j++;
        }
        else
        {
            c[k]=a[i];
            i++;
        }
    }
```



```

        k++;
    }
    if(i>=n)
    {
        while(j<m)
        {
            c[k]=b[j];
            j++;
            k++;
        }
    }
    else if(j>=m)
    {
        while(i<n)
        {
            c[k]=a[i];
            i++;
            k++;
        }
    }
    printf("\n Resultant: ");
    for(k=0;k<n+m;k++)
    {
        printf("%d,",c[k]);
    }
}

```

#### OUTPUT:

```

Enter the total number of elements in 1st array: 3
Enter the total number of elements in 2nd array: 4
Enter the elements of 1st sorted array: 2 5 4

Enter the elements of 2nd sorted array: 1 3 6 7

Resultant: 1,2,3,5,4,6,7,

```

#### RESULT:

The program executed successfully and output is verified.

## PROGRAM-3

25/11/20221

### AIM:

Stack implementation using array.

### PROGRAM:

```
#include <stdio.h>
int stack[100],top,i,n;
void main()
{
    void pop();
    void push();
    void display();
    void peek();
    void exit();
    int choice,i;
    top=-1;
    printf("\n Enter the size of stack: ");
    scanf("%d",&n);
    printf("\n Enter the elements of stack: ");
    for(i=0;i<n;i++)
    {
        top=top+1;
        scanf("%d",&stack[i]);
    }
    do
    {
        printf("\n 1.Display 2.Pop 3.Push 4.Peek 5.Exit");
        printf("\nEnter the choice: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:printf("\nDISPLAY\n");
                    display();
                    break;
            case 2:printf("\nPOP\n");
                    pop();
```

```
        break;
    case 3:printf("\n PUSH\n");
        push();
        break;
    case 4:printf("\n PEEK\n");
        peek();
        break;
    case 5:printf("\n EXIT\n");
        exit(1);
    default:printf("\n Wrong choice");
```

```
    }
}while(choice<=5);
}
```

```
void display()
{
    if(top<0)
    {
        printf("\n Stack is empty");
    }
    else
    {
        for(i=top;i>-1;i--)
        {
            printf("%d,",stack[i]);
        }
    }
}
```

```
void peek()
{
    int data;
    data=stack[top];
    printf("\n %d",data);
```

```

}
void push()
{
    int data;
```

```
printf("enter data to be inserted: ");
scanf("%d",&data);
if(top==n-1)
{
    printf("could not print stack full");
}
else
{
    top=top+1;
    stack[top]=data;

}
}
void pop()
{
    if(top== -1)
    {
        printf("could not delete stack empty");

    }

    else
    {
        printf("Popped element: %d",stack[top]);
        top=top-1;
    }
}
```

## OUTPUT:

```
Enter the size of stack: 2

Enter the elements of stack: 7 4

1.Display 2.Pop 3.Push 4.Peek 5.Exit
Enter the choice: 1

DISPLAY
4,7,
1.Display 2.Pop 3.Push 4.Peek 5.Exit
Enter the choice: 2

POP
Popped element: 4
1.Display 2.Pop 3.Push 4.Peek 5.Exit
Enter the choice: 3

PUSH
enter data to be inserted: 10

1.Display 2.Pop 3.Push 4.Peek 5.Exit
Enter the choice: 4

PEEK

10
1.Display 2.Pop 3.Push 4.Peek 5.Exit
Enter the choice: 
```

## RESULT:

The program executed successfully and output is verified.

## PROGRAM-4

2/11/20221

### AIM:

Implementation of queue using array.

### PROGRAM:

```
#include<stdio.h>
void enqueue();
void dequeue();
void display();
int isfull();
int isempty();
int rear=-1;
int front=-1;
int i,queue[40],data,n;
void main()
{
    int ch;
    printf("\nEnter the size of the array:");
    scanf("%d",&n);
    do{
        printf("\nArray Implementations.");
        printf("\n 1.Enqueue \n 2.Dequeue \n 3.Display");
        printf("\nEnter your choice: ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:enqueue();
                break;
            case 2:dequeue();
                break;
            case 3:display();
                break;
            default:printf("\nInvalid choice");
                break;
        }
    }while(ch<4);
}
```

```

void enqueue()
{
    if(isfull())
    {
        printf("\nQueue is full");
    }
    else
    {
        if(front==-1)
        {
            front=0;
        }
        printf("\nInsert the element: ");
        scanf("%d",&data);
        rear=rear+1;
        queue[rear]=data;
        printf("\n the element inserted is %d",data);
    }
}

```

```

}
void dequeue()
{
    if(isempty())
        printf("\nQueue is empty");
    int data=queue[front];
    front=front+1;
    printf("\nDeleted element is:%d",data);
}

```

```

void display()
{
    if(isempty())
    {
        printf("\nQueue is empty");
    }
    else
    {
        for(i=front;i<=rear;i++)
        {
            printf("%d",queue[i]);
        }
    }
}

```

```
    }  
}  
int isfull()  
{  
    if(rear==n-1)  
    {  
        return 1;  
    }  
    else  
    {  
        return 0;  
    }  
}  
int isempty()  
{  
    if(front<0 || front>rear)  
        return 1;  
    else  
        return 0;  
}
```



## OUTPUT:

```
Enter the size of the array:2
```

```
Array Implementations.
```

```
1.Enqueue
```

```
2.Dequeue
```

```
3.Display
```

```
Enter your choice: 1
```

```
Insert the element: 2
```

```
the element inserted is 2
```

```
Array Implementations.
```

```
1.Enqueue
```

```
2.Dequeue
```

```
3.Display
```

```
Enter your choice: 1
```

```
Insert the element: 6
```

```
the element inserted is 6
```

```
Array Implementations.
```

```
1.Enqueue
```

```
2.Dequeue
```

```
3.Display
```

```
Enter your choice: 1
```

```
Queue is full
```

```
Array Implementations.
```

```
Queue is full
Array Implementations.
 1.Enqueue
 2.Dequeue
 3.Display
Enter your choice: 2

Deleted element is:2
Array Implementations.
 1.Enqueue
 2.Dequeue
 3.Display
Enter your choice: 2

Deleted element is:6
Array Implementations.
 1.Enqueue
 2.Dequeue
 3.Display
Enter your choice: 3

Queue is empty
Array Implementations.
 1.Enqueue
 2.Dequeue
 3.Display
Enter your choice: █
```

## RESULT:

The program executed successfully and output is verified.

## PROGRAM-5

16/11/20221

### AIM:

Implementation of circular queue using array.

### PROGRAM:

```
#include<stdio.h>
void enqueue();
void dequeue();
void display();
int isfull();
int isempty();
int front=-1;
int rear=-1;
int size,x,items[50],i;
void main()
{
    int ch;
    printf("\nEnter the size of the CIRCULAR QUEUE:");
    scanf("%d",&size);
    do
    {
        printf("\nEnter the operations:\n");
        printf("1.Enqueue\n");
        printf("2.Dequeue\n");
        printf("3.Display\n");
        printf("\nEnter your choice:");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:enqueue();
                break;
            case 2:dequeue();
                break;
            case 3:display();
                break;
            default:printf("\nInvalid operation.");
                break;
        }
    }
```

```

    }
}
while(ch<4);
}
void enqueue()
{
    if(isfull())
    {
        printf("\n queue is full");
    }
    else
    {
        if(front==-1)
        {
            front=0;
        }
        rear=(rear+1)% size;
        printf("\n Insert the element: ");
        scanf("%d",&x);
        items[rear]=x;
        printf("\n the element inserted is %d",x);
    }
}
void dequeue()
{
    if(isempty())
    {
        printf("\n queue is empty");
    }
    else
    {
        x=items[front];
        if(front==rear)
        {
            front=-1;
            rear=-1;
        }
    }
}

```

```

        else
        {
            front=(front+1)%size;
        }
        printf("\n deleted element is %d",x);
    }

}

void display()
{
    int i;
    if (isempty())
        printf(" \n Empty Queue\n");
    else {
        for (i = front; i != rear; i = (i + 1) % size) {
            printf("%d ", items[i]);
        }
        printf("%d ", items[i]);
    }
}

int isfull()
{
    if((front==rear+1)||((front==0&&rear==size-1))
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

int isempty()
{
    if(front==-1)
        return 1;
    else
        return 0;
}

```

## OUTPUT:

```
Enter the size of the CIRCULAR QUEUE:3
```

```
Enter the operations:
```

```
1.Enqueue
```

```
2.Dequeue
```

```
3.Display
```

```
Enter your choice:1
```

```
    Insert the element: 2
```

```
    the element inserted is 2
```

```
Enter the operations:
```

```
1.Enqueue
```

```
2.Dequeue
```

```
3.Display
```

```
Enter your choice:1
```

```
    Insert the element: 7
```

```
    the element inserted is 7
```

```
Enter the operations:
```

```
1.Enqueue
```

```
2.Dequeue
```

```
3.Display
```

```
Enter your choice:1
```

```
Insert the element: 4

the element inserted is 4
Enter the operations:
1.Enqueue
2.Dequeue
3.Display

Enter your choice:3
2 7 4
Enter the operations:
1.Enqueue
2.Dequeue
3.Display

Enter your choice:2

deleted element is 2
Enter the operations:
1.Enqueue
2.Dequeue
3.Display

Enter your choice:
```

## RESULT:

The program executed successfully and output is verified.

## PROGRAM-6

6/1/2022

### AIM:

implementation of singly linked list.

### PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
    int value;
    struct node* link;
}node1;
int val,pos,i;
node1* start = NULL;
void in_b();
void in_bw();
void in_end();
void display();
void d_beg();
void d_bw();
void d_end();
void main()
{
    int ch;
    while(1)
    {
        printf("\n-----Linked List Operations-----\n1-Insert node in the
        beginning\n2-Insert node in between\n3-Insert node at the end\n4-Display\n5-Delete
        node from beginning\n6-Delete node in between\n7-Delete node at the
        end\n8-Exit\nEnter your choice : \n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:in_b();
                    break;
            case 2:in_bw();
                    break;
```



```

case 3:in_end();
    break;
case 4:display();
    break;
case 5:d_beg();
    break;
case 6:d_bw();
    break;
case 7:d_end();
    break;
case 8: exit(0);
default:printf("Invalid input");
}
}
}
void in_b()
{
    node1* nptr=malloc(sizeof(node1));
    printf("Enter the element : ");
    scanf("%d",&val);
    nptr->value=val;
    if(start==NULL)
    {
        start=nptr;
        nptr->link=NULL;
    }
    else
    {
        nptr->link=start;
        start=nptr;
    }
    printf("Node inserted\n");
}
void display()
{
    node1* temp;
    if(start==NULL)
    {
        printf("List Empty");
    }
}

```

```

printf("Elements are:\n");
temp=start;
while(temp!=NULL)
{
printf("%d\n",temp->value);
temp=temp->link;
}
}
void in_bw()
{
node1 *temp;
node1* nptr=malloc(sizeof(node1));
printf("Enter the element and position to be inserted :\n");
scanf("%d%d",&val,&pos);
nptr->value=val;
nptr->link=NULL;
temp=start;
if(pos==1)
{
nptr->link=start;
start=nptr;
}
else
{
for(i=1;i<pos-1;i++)
{
temp=temp->link;
}
nptr->link=temp->link;
temp->link=nptr;
}
printf("Node inserted\n");
}
void in_end()
{
node1* temp;
node1* nptr=malloc(sizeof(node1));
printf("Enter the element : ");
scanf("%d",&val);
nptr->value=val;

```

```

    nptr->link=NULL;
    temp=start;
    while(temp->link!=NULL)
    {
        temp=temp->link;

    }
    temp->link=nptr;
    printf("Node inserted\n");
}
void d_beg()
{
    node1* temp;
    if(start==NULL)
        printf("List empty");
    else
    {
        temp=start;
        start=start->link;
        free(temp);
        printf("Node deleted");
    }
}
void d_bw()
{
    node1* temp,*prev;
    printf("Enter position of the node to be deleted ");
    scanf("%d",&pos);
    temp=start;
    if(pos==1)
        start=start->link;
    for(i=1;i<pos;i++)
    {
        prev=temp;
        temp=temp->link;
    }
    prev->link=temp->link;
    free(temp);
    printf("Node deleted");
}

```

```

void d_end()
{
    node1* temp,*prev;
    temp=start;
    while(temp->link!=NULL)
    {
        prev=temp;
        temp=temp->link;
    }
    prev->link=NULL;
    free(temp);
    printf("Node deleted");
}

```

## OUTPUT:

-----Linked List Operations-----

1-Insert node in the beginning

2-Insert node in between

3-Insert node at the end

4-Display

5-Delete node from beginning

6-Delete node in between

7-Delete node at the end

8-Exit

Enter your choice :

1

Enter the element : 4

Node inserted

Enter your choice :

1

Enter the element : 6

Node inserted

Enter your choice :

4

Enter your choice :

4

Elements are:

6

4

Enter your choice :

2

Enter the element and position to be inserted :

8 2

Node inserted

Enter your choice :

4

Elements are:

6

8

4

Enter your choice :

3

Enter the element : 12

Node inserted

Enter your choice :

4

Enter your choice :

4

Elements are:

6

8

4

12

Enter your choice :

5

Node deleted

Enter your choice :

4

Elements are:

8

4

12

## **RESULT:**

The program executed successfully and output is verified.

## PROGRAM-7

13/1/2022

### AIM:

Bst implementation .

### PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>

struct btnode
{
    int value;
    struct btnode *l;
    struct btnode *r;
}*root = NULL, *temp = NULL, *t2, *t1;

void delete1();
void insert();
void delete();
void inorder(struct btnode *t);
void create();
void search(struct btnode *t);
void search1(struct btnode *t,int data);
int smallest(struct btnode *t);

void main()
{
    int ch;

    printf("\nOPERATIONS ---");
    printf("\n1 - Insert an element into tree\n");
    printf("\n2 - Delete an element from the tree\n");
    printf("\n3 - Inorder Traversal\n");
    printf("\n4 - Exit\n");
    while(1)
    {
        printf("\nEnter your choice : ");
```

```

scanf("%d", &ch);
switch (ch)
{
case 1:
    insert();
    break;
case 2:
    delete();
    break;
case 3:
    inorder(root);
    break;
case 4:
    exit(0);
default :
    printf("Wrong choice, Please enter correct choice ");
    break;
}
}
}

```

```

void insert()
{
    create();
    if (root == NULL)
        root = temp;
    else
        search(root);
}

```

```

void create()
{
    int data;

    printf("Enter data of node to be inserted : ");
    scanf("%d", &data);
    temp = (struct btnode *)malloc(1*sizeof(struct btnode));
    temp->value = data;
    temp->l = temp->r = NULL;
}

```



```

void search(struct btnode *t)
{
    if ((temp->value > t->value) && (t->r != NULL))
        search(t->r);
    else if ((temp->value > t->value) && (t->r == NULL))
        t->r = temp;
    else if ((temp->value < t->value) && (t->l != NULL))
        search(t->l);
    else if ((temp->value < t->value) && (t->l == NULL))
        t->l = temp;
}

void inorder(struct btnode *t)
{
    if (root == NULL)
    {
        printf("No elements in a tree to display");
        return;
    }
    if (t->l != NULL)
        inorder(t->l);
    printf("%d -> ", t->value);
    if (t->r != NULL)
        inorder(t->r);
}

void delete()
{
    int data;

    if (root == NULL)
    {
        printf("No elements in a tree to delete");
        return;
    }
    printf("Enter the data to be deleted : ");
    scanf("%d", &data);

    search1(root, data);
}

```

```

void search1(struct btnode *t, int data)
{
    if ((data>t->value))
    {
        t1 = t;
        search1(t->r, data);
    }
    else if ((data < t->value))
    {
        t1 = t;
        search1(t->l, data);
    }
    else if ((data==t->value))
    {
        delete1(t);
    }
}

```

```

void delete1(struct btnode *t)
{
    int k;

    if ((t->l == NULL) && (t->r == NULL))
    {
        if (t1->l == t)
        {
            t1->l = NULL;
        }
        else
        {
            t1->r = NULL;
        }
        t = NULL;
        free(t);
        return;
    }

    else if ((t->r == NULL))
    {

```

```

    if (t1 == t)
    {
        root = t->l;
        t1 = root;
    }
    else if (t1->l == t)
    {
        t1->l = t->l;

    }
    else
    {
        t1->r = t->l;
    }
    t = NULL;
    free(t);
    return;
}

else if (t->l == NULL)
{
    if (t1 == t)
    {
        root = t->r;
        t1 = root;
    }
    else if (t1->r == t)
        t1->r = t->r;
    else
        t1->l = t->r;
    t == NULL;
    free(t);
    return;
}

else if ((t->l != NULL) && (t->r != NULL))
{
    k = smallest(t->r);
    search1(root, k);
    t->value = k;
}

```

```

    }
}

int smallest(struct btnode *t)
{
    if (t->l != NULL)
    {
        return(smallest(t->l));
    }
    else
        return (t->value);
}

```

## OUTPUT:

```

OPERATIONS ---
1 - Insert an element into tree
2 - Delete an element from the tree
3 - Inorder Traversal
4 - Exit

Enter your choice : 1
Enter data of node to be inserted : 3

Enter your choice : 1
Enter data of node to be inserted : 6

Enter your choice : 1
Enter data of node to be inserted : 2

Enter your choice : 3
2 -> 3 -> 6 ->
Enter your choice : 2
Enter the data to be deleted : 3

Enter your choice : 3
2 -> 6 ->
Enter your choice :

```

**RESULT:**

The program executed successfully and output is verified.

## PROGRAM-8

20/1/2022

### AIM:

implementation of doubly linked list.

### PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
    int value;
    struct node* next,*prev;
}node1;
int val,pos,i;
node1* start=NULL;
void in_b();
void in_bw();
void in_end();
void display();
void d_beg();
void d_bw();
void d_end();
void main()
{
    int ch;
    while(1)
    {
        printf("\n-----Doubly Linked List Operations-----\n1-Insert node in the
beginning\n2-Insert node in between\n3-Insert node at the end\n4-Display\n5-Delete
node from beginning\n6-Delete node in between\n7-Delete node at the
end\n8-Exit\nEnter your choice : \n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:in_b();
                break;
            case 2:in_bw();
                break;
```

```

        case 3:in_end();
            break;
        case 4:display();
            break;
        case 5:d_beg();
            break;
        case 6:d_bw();
            break;
        case 7:d_end();
            break;
        case 8:exit(0);
        default:printf("Invalid input");
    }
}
}
void in_b()
{
    node1* nptr=malloc(sizeof(node1));
    if(nptr==NULL)
    {
        printf("Memory overflow\n");
    }
    else
    {
        printf("Enter the element : ");
        scanf("%d",&val);
        nptr->value=val;
        if(start==NULL)
        {
            start=nptr;
            nptr->next=NULL;
            nptr->prev=NULL;
        }
        else
        {
            nptr->next=start;
            nptr->prev=NULL;
            start->prev=nptr;
            start=nptr;
        }
    }
}

```

```

    }
    printf("Node inserted\n");
}
void in_bw()
{
    int c=1;
    node1* ptr;
    node1 *temp=start;
    node1* nptr=malloc(sizeof(node1));
    printf("Enter the element to be inserted : ");
    scanf("%d",&val);
    nptr->value=val;
    ptr=temp;
    nptr->next=NULL;
    printf("\nEnter position : ");
    scanf("%d",&pos);
    if(temp==NULL)
    {
        start=nptr;
        nptr->next=NULL;
        nptr->prev=NULL;
    }
    else
    {
        if(pos==1)
        {
            nptr->next=start;
            start->prev=nptr;
            start=nptr;
        }
        else
        {
            while(temp!=NULL)
            {
                if(c==pos)
                {
                    nptr->next=ptr->next;
                    ptr->next->prev=nptr;
                    nptr->prev=ptr;
                    ptr->next=nptr;
                }
            }
        }
    }
}

```



```

        break;
    }
    ptr=temp;
    temp=temp->next;
    c++;
}
}
}
printf("Node inserted\n");
}
void in_end()
{
    node1* ptr;
    node1* temp=start;
    node1* nptr=malloc(sizeof(node1));
    if(nptr==NULL)
    {
        printf("Memory Overflow\n");
    }
    else
    {
        printf("Enter the element : ");
        scanf("%d",&val);
        nptr->value=val;
        while(temp!=NULL)
        {
            ptr=temp;
            temp=temp->next;
        }
        ptr->next=nptr;
        nptr->next=NULL;
        nptr->prev=ptr;
        printf("Node inserted\n");
    }
}
void display()
{
    node1* temp;
    if(start==NULL)
    {

```

```

        printf("List Empty");
    }
    else
    {
        printf("Elements are:\n");
        temp=start;
        while(temp!=NULL)
        {
            printf("%d\t",temp->value);
            temp=temp->next;
        }
    }
}

void d_beg()
{
    node1* temp;
    if(start==NULL)
    {
        printf("List empty");
    }
    else if(start->next==NULL)
    {
        printf("Node deleted is %d ",start->value);
        start=NULL;
        free(start);
    }
    else
    {
        temp=start;
        start=start->next;
        printf("Node deleted is %d",temp->value);
        temp->next->prev=NULL;
        free(temp);
    }
}

void d_bw()
{
    int c=1,f=0;

```

```

node1* temp,*nptr;
if(start==NULL)
{
    printf("List empty\n");
}
else
{
    printf("Enter position of the node to be deleted : ");
    scanf("%d",&pos);
    temp=start;
    nptr=start;
    if(pos==1)
    {
        if(start->next==NULL)
        {
            printf("Node deleted is %d ",start->value);
            start=NULL;
            free(start);
            f=1;
        }
        else
        {
            temp=start;
            start=start->next;
            printf("Node deleted is %d",temp->value);
            temp->next->prev=NULL;
            free(temp);
            f=1;
        }
    }
    else
    {
        while(temp!=NULL)
        {
            if(c==pos)
            {
                nptr->next=temp->next;
                printf("\nElement deleted : %d",temp->value);
                temp->next->prev=nptr;
            }
        }
    }
}

```

```

        free(temp);
        f=1;
    }
    nptr=temp;
    temp=temp->next;
    c++;
}
}
if(f==0)
{
    printf("Invalid position\n");
}
}
}
void d_end()
{
    node1* temp,*ptr;
    int c=1;
    if(start==NULL)
    {
        printf("\nEmpty List");
    }
    else
    {
        temp=start;
        while(temp->next!=NULL)
        {
            ptr=temp;
            temp=temp->next;
            c++;
        }
        printf("\nElement Deleted : %d",temp->value);
        ptr->next=NULL;
        free(temp);
        if (c==1)
            start=NULL;
    }
}
}

```

## OUTPUT:

```
-----Doubly Linked List Operations-----
1-Insert node in the beginning
2-Insert node in between
3-Insert node at the end
4-Display
5-Delete node from beginning
6-Delete node in between
7-Delete node at the end
8-Exit
Enter your choice :
1
Enter the element : 4
Node inserted

-----Doubly Linked List Operations-----
1-Insert node in the beginning
2-Insert node in between
3-Insert node at the end
4-Display
5-Delete node from beginning
6-Delete node in between
7-Delete node at the end
8-Exit
Enter your choice :
1
Enter the element : 7
Node inserted
```

```
-----Doubly Linked List Operations-----
1-Insert node in the beginning
2-Insert node in between
3-Insert node at the end
4-Display
5-Delete node from beginning
6-Delete node in between
7-Delete node at the end
8-Exit
Enter your choice :
1
Enter the element : 3
Node inserted

-----Doubly Linked List Operations-----
1-Insert node in the beginning
2-Insert node in between
3-Insert node at the end
4-Display
5-Delete node from beginning
6-Delete node in between
7-Delete node at the end
8-Exit
Enter your choice :
2
Enter the element to be inserted : 10

Enter position : 2
Node inserted
```

```
-----Doubly Linked List Operations-----
1-Insert node in the beginning
2-Insert node in between
3-Insert node at the end
4-Display
5-Delete node from beginning
6-Delete node in between
7-Delete node at the end
8-Exit
Enter your choice :
4
Elements are:
3      10      7      4
-----Doubly Linked List Operations-----
1-Insert node in the beginning
2-Insert node in between
3-Insert node at the end
4-Display
5-Delete node from beginning
6-Delete node in between
7-Delete node at the end
8-Exit
Enter your choice :
3
Enter the element : 11
Node inserted
```

```
-----Doubly Linked List Operations-----
1-Insert node in the beginning
2-Insert node in between
3-Insert node at the end
4-Display
5-Delete node from beginning
6-Delete node in between
7-Delete node at the end
8-Exit
Enter your choice :
4
Elements are:
3      10      7      4      11
-----Doubly Linked List Operations-----
1-Insert node in the beginning
2-Insert node in between
3-Insert node at the end
4-Display
5-Delete node from beginning
6-Delete node in between
7-Delete node at the end
8-Exit
Enter your choice :
5
Node deleted is 3
```



```
-----Doubly Linked List Operations-----
1-Insert node in the beginning
2-Insert node in between
3-Insert node at the end
4-Display
5-Delete node from beginning
6-Delete node in between
7-Delete node at the end
8-Exit
Enter your choice :
6
Enter position of the node to be deleted : 3

Element deleted : 4
-----Doubly Linked List Operations-----
1-Insert node in the beginning
2-Insert node in between
3-Insert node at the end
4-Display
5-Delete node from beginning
6-Delete node in between
7-Delete node at the end
8-Exit
Enter your choice :
7

Element Deleted : 11
```

## RESULT:

The program executed successfully and output is verified.

## PROGRAM-9

20/1/2022

### AIM:

Implementation of bit string operations.

### PROGRAM:

```
#include <stdio.h>
void input();
void output();
void setunion();
void intersection();
void difference();
int a[]={0,0,0,0,0,0,0,0,0,0},b[]={0,0,0,0,0,0,0,0,0,0} ;
int main()
{
    int ch,wish;

    do
    {
        printf("\n____MENU____\n");
        printf("1.Input\n2.Union\n3.Intersection\n4.Difference\n");
        printf("enter choice\n");
        scanf("%d",&ch);

        switch(ch)
        {
            case 1:input();
                    break;
            case 2:setunion();
                    break;
            case 3:intersection();
                    break;
            case 4:difference();
                    break;

        }

        printf("\nDo you wish to continue?(1/0)\n");
        scanf("%d",&wish);
```

```

    }while(wish==1);

}

void input()
{ int n,x,i;
  printf("enter size of the 1st set\n");
  scanf("%d",&n);
  printf("enter elements\n");
  for(i=0;i<n;i++)
  { scanf("%d",&x);
    a[x-1]=1;
  }
  printf("enter size of the 2nd set\n");
  scanf("%d",&n);
  printf("enter elements\n");
  for(i=0;i<n;i++)
  { scanf("%d",&x);
    b[x-1]=1;
  }
  printf("\n1st set\n");
  for(i=0;i<9;i++)
  { printf("%d",a[i]);
  }
  printf("\n2nd set\n");
  for(i=0;i<9;i++)
  { printf("%d",b[i]);
  }
}

void output(int c[])
{ int i;
  printf("\n Set is");
  for(i=0;i<9;i++)
  { if (c[i]!=0)
    printf(" %d ",i+1);

  }
}

```

## OUTPUT:

```
____MENU____
1.Input
2.Union
3.Intersection
4.Difference
enter choice
1
enter size of the 1st set
5
enter elements
1 2 3 4 5
enter size of the 2nd set
3
enter elements
4 5 6

1st set
111110000
2nd set
000111000
Do you wish to continue ?(1/0)
1

____MENU____
1.Input
2.Union
3.Intersection
4.Difference
```

```
____MENU____
1.Input
2.Union
3.Intersection
4.Difference
enter choice
2
111111000
Set is 1 2 3 4 5 6
Do you wish to continue ?(1/0)
1
```

```
____MENU____
1.Input
2.Union
3.Intersection
4.Difference
enter choice
3
000110000
Set is 4 5
Do you wish to continue ?(1/0)
1
```

```
____MENU____
1.Input
2.Union
3.Intersection
4.Difference
```

```
____MENU____
1.Input
2.Union
3.Intersection
4.Difference
enter choice
4
111000000
Set is 1 2 3
Do you wish to continue ?(1/0)

```

## RESULT:

The program executed successfully and output is verified.

## PROGRAM-10

20/1/2022

### AIM:

Implementation of disjoint set.

### PROGRAM:

```
#include <stdio.h>
struct DisjSet {
    int parent[10];
    int rank[10];
    int n;
}dis;
void makeSet()
{
    for (int i = 0; i < dis.n; i++) {
        dis.parent[i] = i;
        dis.rank[i]=0;
    }
}
void displaySet()
{ printf("\nParent Array\n");
  for (int i = 0; i < dis.n; i++) {
      printf("%d ",dis.parent[i]); }
  printf("\nRank Array\n");
  for (int i = 0; i < dis.n; i++)
  {
      printf("%d ",dis.rank[i]);
  }
  printf("\n");
}
int find(int x)
{
    if (dis.parent[x] != x)
    {
        dis.parent[x] = find(dis.parent[x]);
    }
    return dis.parent[x];
}
```

```

void Union(int x, int y)
{
    int xset = find(x);
    int yset = find(y);
    if (xset == yset)
        return;
    if (dis.rank[xset] < dis.rank[yset]) {
        dis.parent[xset] = yset;
        dis.rank[xset] = -1;
    }
    else if (dis.rank[xset] > dis.rank[yset]) {
        dis.parent[yset] = xset;
        dis.rank[yset] = -1;
    }
    else {
        dis.parent[yset] = xset;
        dis.rank[xset] = dis.rank[xset] + 1;
        dis.rank[yset] = -1;
    }
}

```

```

int main()
{
    int n,x,y;
    printf("How many elements ?");
    scanf("%d",&dis.n);
    makeSet();
    int ch,wish;

```

```

do
{
    printf("\n____MENU____\n");
    printf("1. Union \n2.Find\n3.Display\n");
    printf("enter choice\n");
    scanf("%d",&ch);

```

```

switch(ch)
{
    case 1: printf("Enter elements to perform union");
            scanf("%d %d",&x,&y);
            Union(x, y);

```

```

        break;
    case 2: printf("Enter elements to check if connected components");
        scanf("%d %d",&x,&y);
        if (find(x) == find(y))
            printf("Connected components\n") ;
        else
            printf("Not onnected components \n") ;
        break;
    case 3: displaySet();
        break;
}
printf("\nDo you wish to continue ?(1/0)\n");
scanf("%d",&wish);
}while(wish==1);
return 0;
}

```



## OUTPUT:

```
Do you wish to continue ?(1/0)
1

____MENU____
1. Union
2.Find
3.Display
enter choice
2
Enter elements to check if connected components34
34
Connected components

Do you wish to continue ?(1/0)
1

____MENU____
1. Union
2.Find
3.Display
enter choice
3

Parent Array
0 1 2 3
Rank Array
0 0 0 0

Do you wish to continue ?(1/0)
```

## RESULT :

The program executed successfully and output is verified.

## PROGRAM-11

2/3/2022

### AIM:

Implementation of linked stack

### PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
void push();
void pop();
void display();
struct node
{
int val;
struct node *next;
};
struct node *start;

void main ()
{
int choice=0;
printf("\nStack operations using linked list\n");
while(choice != 4)
{
printf("\n-----Menu-----\n");
printf("\n1.Push\n2.Pop\n3.Display\n4.Exit");
printf("\n Enter your choice : \n");
scanf("%d",&choice);
switch(choice)
{
case 1:
{
push();
break;
}
case 2:
{
pop();
```

```

        break;
    }
    case 3:
    {
        display();
        break;
    }
    case 4:
    {
        printf("Exiting....");
        break;
    }
    default:
    {
        printf("Please Enter valid choice ");
    }
};
}
}
void push ()
{
    int val;
    struct node *ptr = (struct node*)malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("Not able to push the element");
    }
    else
    {
        printf("Enter the value");
        scanf("%d",&val);
        if(start==NULL)
        {
            ptr->val = val;
            ptr -> next = NULL;
            start=ptr;
        }
        else
        {
            ptr->val = val;

```

```

        ptr->next = start;
        start=ptr;

    }
    printf("Item pushed");

}

}

void pop()
{
    int item;
    struct node *ptr;
    if (start == NULL)
    {
        printf("Underflow");
    }
    else
    {
        item = start->val;
        ptr = start;
        start = start->next;
        free(ptr);
        printf("Item popped");

    }
}

void display()
{
    int i;
    struct node *ptr;
    ptr=start;
    if(ptr == NULL)
    {
        printf("Stack is empty\n");
    }
    else
    {
        printf("Printing Stack elements \n");
        while(ptr!=NULL)

```

```

        {
            printf("%d\n",ptr->val);
            ptr = ptr->next;
        }
    }
}

```

## OUTPUT :

```

Stack operations using linked list

-----Menu-----

1.Push
2.Pop
3.Display
4.Exit
Enter your choice :
1
Enter the value 3
Item pushed
-----Menu-----

1.Push
2.Pop
3.Display
4.Exit
Enter your choice :
1
Enter the value 7
Item pushed
-----Menu-----

1.Push
2.Pop
3.Display
4.Exit
Enter your choice :
1
Enter the value10
Item pushed

```

```
-----Menu-----
1.Push
2.Pop
3.Display
4.Exit
Enter your choice :
3
Printing Stack elements
10
7
3
-----Menu-----
1.Push
2.Pop
3.Display
4.Exit
Enter your choice :
2
Item popped
-----Menu-----
1.Push
2.Pop
3.Display
4.Exit
Enter your choice :
```

## RESULT :

The program executed successfully and output is verified.

## PROGRAM-12

2/3/2022

### AIM:

BFS and DFS

### PROGRAM:

```
#include<stdio.h>
#include<conio.h>
int q[20],top=-1,front=-1,rear=-1,a[20][20],vis[20],stack[20];
void add(int item);
void bfs(int s,int n);
void dfs(int s,int n);
void push(int item);
int pop();
int delete1();
int main()
{
    int n,i,s,ch,j;
    char c,dummy;
    printf("ENTER THE NUMBER VERTICES ");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            printf(" %d %d ",i,j);
            scanf("%d",&a[i][j]);
        }
    }
    printf("THE 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("ENTER THE SOURCE VERTEX :");
scanf("%d",&s);

switch(ch)
{
case 1:bfs(s,n);
break;
case 2:
dfs(s,n);
break;
}
printf("DO U WANT TO CONTINUE(Y/N) ? ");
scanf("%c",&dummy);
scanf("%c",&c);
}while((c=='y')||(c=='Y'));
return 0;
}

void bfs(int s,int n)
{
int p,i;
add(s);
vis[s]=1;
p= delete1();
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=delete1();
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 delete1()
{
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 3  
1 1 2  
1 2 4  
1 3 7  
2 1 2  
2 2 1  
2 3 9  
3 1 5  
3 2 6  
3 3 11  
THE ADJACENCY MATRIX IS  
2 4 7  
2 1 9  
5 6 11  
  
MENU  
1.B.F.S  
2.D.F.S  
ENTER YOUR CHOICE 1  
ENTER THE SOURCE VERTEX :2  
2 1 3 DO U WANT TO CONTINUE(Y/N) ? y  
  
MENU  
1.B.F.S  
2.D.F.S  
ENTER YOUR CHOICE2  
ENTER THE SOURCE VERTEX :1  
1 3 2 DO U WANT TO CONTINUE(Y/N) ?
```

### RESULT :

The program executed successfully and output is verified.

## PROGRAM-13

2/3/2022

### AIM:

Program to find minimum cost spanning tree using Prim's algorithm

### PROGRAM:

```
#include<stdio.h>
#include<conio.h>
int n,cost[10][10];
void prim();
void main()
{
    int i,j;
    printf("\nEnter the number of vertices:");
    scanf("%d",&n);
    printf("\nEnter the costs of edges in matrix form:");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            scanf("%d",&cost[i][j]);
        }
    }
    printf("\n The matrix is:\n");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            printf("%d ",cost[i][j]);
        }
        printf("\n\n");
    }
    prim();
    getch();
}

void prim()
{

```

```

int i,j,k,l,x,nr[10],temp,mincost=0,tree[10][3];
temp=cost[0][0];
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
if(temp>cost[i][j])
{
temp=cost[i][j];
k=i;
l=j;
}
}
tree[0][0]=k;
tree[0][1]=l;
tree[0][2]=temp;
mincost=temp;
for(i=0;i<n;i++)
{
if(cost[i][k]<cost[i][l])
nr[i]=k;
else
nr[i]=l;
}
nr[k]=100;
nr[l]=100;
temp=99;
for(i=1;i<n-1;i++)
{
for(j=0;j<n;j++)
{
if(nr[j]!=100&&cost[j][nr[j]]<temp)
{
temp=cost[j][nr[j]];
x=j;
}
}
tree[i][0]=x;
tree[i][1]=nr[x];

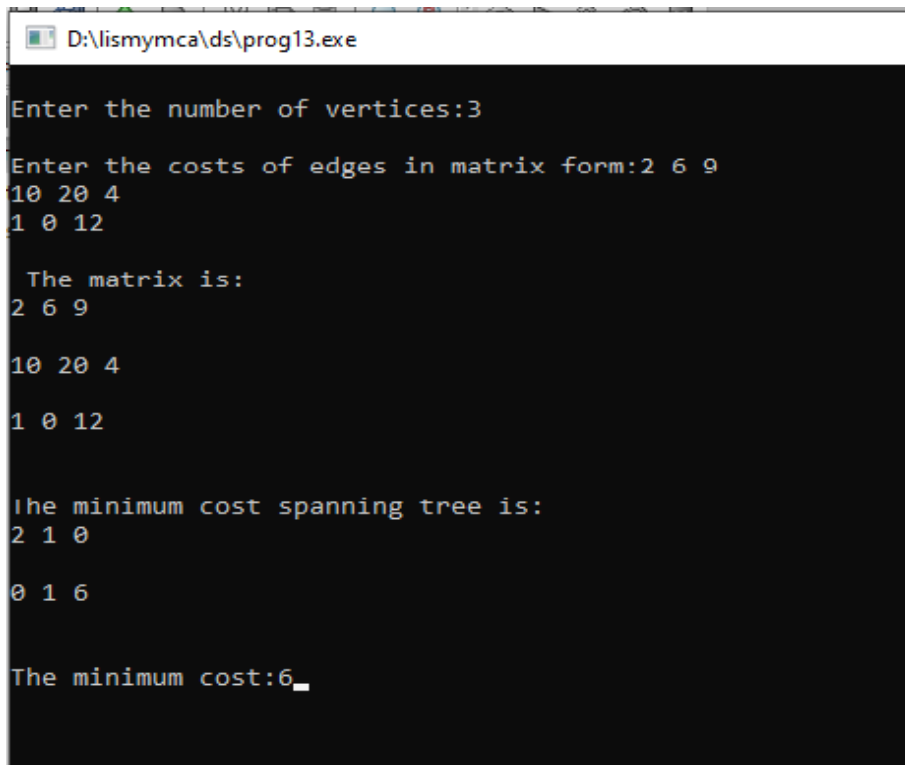
```

```

    tree[i][2]=cost[x][nr[x]];
    mincost=mincost+cost[x][nr[x]];
    nr[x]=100;
    for(j=0;j<n;j++)
    {
if(nr[j]!=100&&cost[j][nr[j]]>cost[j][x])
nr[j]=x;
    }
    temp=99;
    }
    printf("\nThe minimum cost spanning tree is:\n");
    for(i=0;i<n-1;i++)
    {
for(j=0;j<3;j++)
{
printf("%d ",tree[i][j]);
}
printf("\n\n");
    }
    printf("\nThe minimum cost:%d",mincost);
}

```

### OUTPUT :



```

D:\lismymca\ds\prog13.exe

Enter the number of vertices:3

Enter the costs of edges in matrix form:2 6 9
10 20 4
1 0 12

The matrix is:
2 6 9

10 20 4

1 0 12

The minimum cost spanning tree is:
2 1 0

0 1 6

The minimum cost:6_

```

**RESULT :**

The program executed successfully and output is verified.

## PROGRAM-14

2/3/2022

### AIM:

Program to find minimum cost spanning tree using Kruskals Algorithm

### PROGRAM:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int i,j,k,a,b,u,v,n,ne=1;
int min,mincost=0,cost[9][9],parent[9];
int find(int);
int uni(int,int);
void main()
{
    printf("\nEnter the no. of vertices:");
    scanf("%d",&n);
    printf("\nEnter the cost adjacency matrix:\n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&cost[i][j]);
            if(cost[i][j]==0)
                cost[i][j]=999;
        }
    }
    printf("The edges of Minimum Cost Spanning Tree are\n");
    while(ne < n)
    {
        for(i=1,min=999;i<=n;i++)
        {
            for(j=1;j <= n;j++)
            {
                if(cost[i][j] < min)
                {
                    min=cost[i][j];
                    a=u=i;
```



```

        b=v=j;
    }
    }
    }
    u=find(u);
    v=find(v);
    if(uni(u,v))
    {
        printf("%d edge (%d,%d) =%d\n",ne++,a,b,min);
        mincost +=min;
    }
    cost[a][b]=cost[b][a]=999;
    }
    printf("\n\tMinimum cost = %d\n",mincost);
    getch();
}
int find(int i)
{
    while(parent[i])
        i=parent[i];
    return i;
}
int uni(int i,int j)
{
    if(i!=j)
    {
        parent[j]=i;
        return 1;
    }
    return 0;
}

```

## OUTPUT :

```
Enter the no. of vertices:3

Enter the cost adjacency matrix:
1 2 3
7 4 1
9 8 5
The edges of Minimum Cost Spanning Tree are
1 edge (2,3) =1
2 edge (1,2) =2

      Minimum cost = 3
```

## RESULT :

The program executed successfully and output is verified.