FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)™

HORMIS NAGAR, MOOKKANNOOR, ANGAMALY-683577

FOCUS ON EXCELLENCE

20MCA135 DATA STRUCTURE LAB

LABORATORY RECORD

Name: ELIZABA MARIYAM BINNY

Branch: MASTER OF COMPUTER APPLICATIONS
Semester: 1 **Batch: B** **Roll No: 53**

Register Number: ***FIT21MCA-2053***

MARCH 2022

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)™

HORMIS NAGAR, MOOKKANNOOR, ANGAMALY-683577

FOCUS ON EXCELLENCE

CERTIFICATE

This is to certify that this is a Bonafide record of the Practical work done by ***Elizaba Mariyam Binny(FIT21MCA-2053)*** *in the* ***20MCA135 DATA STRUCTURE LAB*** *Laboratory towards the partial fulfilment for the award of the Master Of Computer Applications during the academic year 2021-2022.*

Signature of Staff in Charge

Name:

Signature of H O D

Name:

Date of University practical examination ……

Signature of
Internal Examiner

Signature of
External Examiner

CONTENTS

PROGRAM 1

AIM : Write a program to print maximum and minimum in an array.

INPUT

```
#include<stdio.h>
void main()
{int i,n;
    printf("enter array size\n");
    scanf("%d",&n);
    int numbers[n];
    printf("enter array elements\n");
        for(i=0;i<n;i++)
        {
            scanf("%d",&numbers[i]);
        }
        int max=0;
    int min=9999999;
        for(i=0;i<n;i++)
        {
            if(numbers[i]<min)
                {
                min=numbers[i];
                }
                else if(numbers[i]>max)
                {
                max=numbers[i];
                }
        }
    printf("smallest elements is %d\n",min);
    printf("largest elements is %d\n",max);
}
```

OUTPUT

```
stud@debian:~/elizabamca/ds/ADS$ gcc armaxmin.c
stud@debian:~/elizabamca/ds/ADS$ ./a.out
enter array size
5
enter array elements
1 4 7 3 9
smallest elements is 1
largest elements is 9
```

PROGRAM 2

AIM : Write a program to merge two sorted arrays

INPUT

```
#include <stdio.h>
void main()
{    int n1,n2,n3;
   printf("\nEnter the size of first array ");
   scanf("%d",&n1);
   printf("\nEnter the size of second array ");
   scanf("%d",&n2);
   n3=n1+n2;
   printf("\nEnter the sorted array elements");
   int a[n1],b[n2],c[n3];
   for(int i=0;i<n1;i++)
   {  scanf("%d",&a[i]);
     c[i]=a[i];
   }
   int k=n1;
   printf("\nEnter the sorted array elements");
   for(int i=0;i<n2;i++)
   {  scanf("%d",&b[i]);
      c[k]=b[i];
      k++;
   }
   printf("\nThe merged array..\n");
   for(int i=0;i<n3;i++)
   printf("%d ",c[i]);
}
```

OUTPUT

```
stud@debian:~/elizabamca/ds/ADS$ gcc mergesort.c
stud@debian:~/elizabamca/ds/ADS$ ./a.out

Enter the size of first array 4

Enter the size of second array 3

Enter the sorted array elements 2 3 4 5

Enter the sorted array elements6 7 8

The merged array..
2 3 4 5 6 7 8 stud@debian:~/elizabamca/ds/ADS$
```

PROGRAM 3

AIM : Write a program to implement stack using array.

INPUT

```
#include<stdio.h>
#include<stdlib.h>
int stack[8],value,length=8,top=-1,choice;
void display()
    {       if(top==-1)
            {       printf("stack empty\n");
            }
            else
            {printf("status \n");
            for(int i=0;i<=top;i++)
            {printf("%d ",stack[i]);
            }
            }
            printf("\n\n");
    }
void push()
    {       if(top==length)
            {       printf("stack full\n");
            }
            else
            {       printf("Enter the value to push\n");
                    scanf("%d",&value);
                    top++;
                    stack[top]=value;
                    printf("pushed\n");
                    display();
            }
```

```
    }
void pop()
    {       if(top==-1)
            {       printf("stack empty\n");
            }
            else
            {       printf("poped %d\n",stack[top]);
                    top--;
            }
                    display();
    }
void main()
    {       while(1)
            {printf("1.PUSH \n2.POP \n3.DISPLAY \n4.EXIT \nEnter Choice ");
                    scanf("%d",&choice);
                    switch(choice)
                    {
                            case 1:push();
                            break;
                            case 2:pop();
                            break;
                            case 3:display();
                            break;
                            case 4:exit(1);
                            break;
                            default: printf("wrong choice,%d is not
valid\n",choice);
                            break;
                    }
            }
    }
```

OUTPUT

```
stud@debian:~/elizabamca/ds/ADS$ gcc switch.c
stud@debian:~/elizabamca/ds/ADS$ ./a.out
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter Choice 1
Enter the value to push
2
pushed
status
2

1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter Choice 1
Enter the value to push
3
pushed
status
2 3

1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter Choice 2
poped 3
status
2

1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter Choice
```

PROGRAM 4

AIM : Write a program to implement queue using array.

INPUT

```
#include<stdio.h>
#include<stdlib.h>
int front=-1,rear=-1,queue[3],size=3,value;
void display()
{    printf("status\n");
   if(front==-1||front>rear)
   {  printf("stack empty\n");
   }
   for(int i=front;i<=rear;i++)
   {        printf("%d\n",queue[i]);
   }
   printf("\n\n");
}
void enqueue()
{    if(rear==size-1)
   {       printf("full queue\n");
   }
   else
   { if(front==-1)
      {    front=0;
      }
      printf("enter the element to be added\n");
      scanf("%d",&value);
      rear++;
      queue[rear]=value;
   }
   display();
```

```
}
void dequeue()
{    if(front==-1||front>rear)
   {       printf("queue empty");
   }    else
   {   printf("status\n removed %d\n",queue[front]);
       front++;
   }
   display();
}
void main()
{    while(1)
   { int choice;
       printf("enter choice 1.enqueue 2.dequeue 3.display 4.exit\n");
       scanf("%d",&choice);
       switch(choice)
       {   case 1: enqueue();
          break;
          case 2: dequeue();
          break;
          case 3: display();
          break;
          case 4: exit(0);
          break;
          default:printf("wrong choice");
          break;
       }
   }
}
```

OUTPUT

```
stud@debian:~/elizabamca/ds/ADS$ gcc queue1.c
stud@debian:~/elizabamca/ds/ADS$ ./a.out
enter choice 1.enqueue 2.dequeue 3.display 4.exit
1
enter the element to be added
3
status
3


enter choice 1.enqueue 2.dequeue 3.display 4.exit
1
enter the element to be added
4
status
3
4


enter choice 1.enqueue 2.dequeue 3.display 4.exit
2
status
 removed 3
status
4


enter choice 1.enqueue 2.dequeue 3.display 4.exit
3
status
4


enter choice 1.enqueue 2.dequeue 3.display 4.exit
```

PROGRAM 5

AIM : Write a program to implement circular queue.

INPUT

```
#include<stdio.h>
#include<stdlib.h>
int cqueue1[3],front=-1,rear=-1,size=3,value;
void display()
{
printf("status \n");
if(rear>=front)
{
   for(int i=front;i<=rear;i++)
   {
      printf("%d ",cqueue1[i]);
   }
}
else
   {
       for (int i = front; i < size; i++)
           printf("%d ", cqueue1[i]);

       for (int i = 0; i <= rear; i++)
           printf("%d ", cqueue1[i]);
   }
printf("\n\n");
}
void enqueue()
{
      if ((front==0 && rear==size-1)||(rear==front-1))
   {
```

```
        printf("cqueue full");
    }
    else
    {
      if(front==-1)
      {
            front++;
      }
            printf("enter value ");
            scanf("%d",&value);
            rear=(rear+1)%size;
            cqueue1[rear]=value;
    }
display();
}
void dequeue()
{
      if (front==-1||rear==-1)
      {
            printf("cqueue empty");
      }
      else
      {
            if (front==rear)
            {
                  printf("removed %d\n",cqueue1[front]);
                  front=-1;
                  rear=-1;
            }
            else
```

```
            {
                printf("removed %d\n",cqueue1[front]);
                front=(front+1)%size;
            }
        }
display();
}
void main()
    {
        while(1)
        {int choice;
            printf("1.enqueue \n2.dequeue \n3.DISPLAY \n4.EXIT \nEnter
Choice ");
            scanf("%d",&choice);
            switch(choice)
            {
                case 1:enqueue();
                break;
                case 2:dequeue();
                break;
                case 3:display();
                break;
                case 4:exit(1);
                break;
                default: printf("wrong choice,%d is not
valid\n",choice);
                break;
            }
        }
}
```

OUTP

```
stud@debian:~/elizabamca/ds/ADS$ gcc cqueue.c
stud@debian:~/elizabamca/ds/ADS$ ./a.out
1.enqueue
2.dequeue
3.DISPLAY
4.EXIT
Enter Choice 1
enter value 2
status
2

1.enqueue
2.dequeue
3.DISPLAY
4.EXIT
Enter Choice 1
enter value 3
status
2 3

1.enqueue
2.dequeue
3.DISPLAY
4.EXIT
Enter Choice 1
enter value 4
status
2 3 4

1.enqueue
2.dequeue
3.DISPLAY
4.EXIT
Enter Choice 2
removed 2
status
3 4
```

PROGRAM 6

AIM : Write a program to implement singly linked list.

INPUT

```
#include<stdio.h>
#include<stdlib.h>
struct node *l,*new,*ptr,*tm;
struct node
{    int data;
     struct node *next;
};
struct node *head=NULL;
void create()
{    int value;
     new=(struct node*)malloc(sizeof(struct node));
     printf("enter value to insert:\n");
     scanf("%d",&value);
     new->data=value;
     new->next=NULL;
}
void display()
{    if(head==NULL)
     {        printf("empty");
     }
     else
     {         l=head;
               while(l!=NULL)
               {         printf(" %d ",l->data);
                         l=l->next;
               }
     }
```

```
        printf("\n\n");
}
void insert_beg()
{       create();
        if(head==NULL)
        {       head=new;
        }
        else
        {       new->next=head;
                head=new;
        }
        printf("display: ");
display();
}
void insert_end()
{       create();
        if(head==NULL)
        {       head=new;
        }
        else
        {       l=head;
                while(l->next!=NULL)
                {       l=l->next;
                }
                l->next=new;
        }
        printf("display: ");
display();
}
void insert_pos()
```

```
{     create();
      if(head==NULL)
      {       printf("empty");
              head=new;
      }
      else
      {       l=head;
              int pos,i;
              printf("enter position to insert:\n");
              scanf("%d",&pos);
              if(pos==0)
                      {       insert_beg();
                      }
              else
                      for(i=0;i<pos-1;i++)
                      {       l=l->next;
                              if(head==NULL)
                              {       printf("error");
                              }
                      }
                      new->next=l->next;
                      l->next=new;
      }
      printf("display: ");
      display();
}
void delete_beg()
{   if(head==NULL)
  {    printf("empty");
  }
```

```
        else
        {       if(head->next==NULL)
                {       ptr=head;
                        head=NULL;
                        printf("removed: %d\n",ptr->data);
                        free(ptr);
                }
                else
                {       ptr=head;
                        head=head->next;
                        printf("removed: %d\n",ptr->data);
                        free(ptr);
                }
        }
printf("display: ");
display();
}
void delete_end()
{   if(head==NULL)
  {
    printf("empty");
  }   else
        {       if(head->next==NULL)
                {       ptr=head;
                        head=NULL;
                        printf("removed: %d\n",ptr->data);
                        free(ptr);
                }else
                {       ptr=head;
                        while(ptr->next!=NULL)
```

```
                    {       tm=ptr;
                            ptr=ptr->next;
                    }tm->next=NULL;
                    printf("removed: %d\n",ptr->data);
                    free(ptr);
            }
    }
printf("display: ");
display();
}   void delete_pos()
{   if(head==NULL)
    {printf("empty");
    }       else
    {int pos;
    printf("enter position \n");
    scanf("%d",&pos);
            if(pos==0)
            {delete_beg();
            }else
            {       ptr=head;
                    for(int i=0;i<pos;i++)
                    {while(ptr->next!=NULL)
                            {       tm=ptr;
                                    ptr=ptr->next;
                            }
                    }
                    tm->next=ptr->next;
                    printf("removed: %d\n",ptr->data);
                    free(ptr);}}
printf("display: ");
```

```
display();
}   void main()
{   int choice; printf("1.insert at beggining \n2.insert at end \n3.insert at position
\n4.delete at beginning \n5.delete at end \n6.display  \n7.delete at position \n8.exit
\n");
    while(1)
    {       printf("Enter Choice: \n");
            scanf("%d",&choice);
            switch(choice)
            {       case 1:insert_beg();
                    break;
                    case 2:insert_end();
                    break;
                    case 3:insert_pos();
                    break;
                    case 4:delete_beg();
                    break;
                    case 5:delete_end();
                    break;
                    case 6:printf("status: ");
                            display();
                    break;
                    case 7:delete_pos();
                    break;
                    case 8:exit(0);
                    break;
                    default: printf("wrong choice,%d is not valid\n",choice);
                    break;
            }
    }
}
```

OUTPUT

```
^C
stud@debian:~/elizabamca/ds$ gcc linli.c
stud@debian:~/elizabamca/ds$ ./a.out
1.insert at beggining
2.insert at end
3.insert at position
4.delete at beginning
5.delete at end
6.display
7.delete at position
8.exit
Enter Choice:
1
enter value to insert:
3
display:  3

Enter Choice:
1
enter value to insert:
4
display:  4  3

Enter Choice:
2
enter value to insert:
5
display:  4  3  5

Enter Choice:
3
enter value to insert:
6
enter position to insert:
2
display:  4  3  6  5

Enter Choice:
1

Enter Choice:
1
enter value to insert:
7
display:  7  4  3  6  5

Enter Choice:
4
removed: 7
display:  4  3  6  5

Enter Choice:
5
removed: 5
display:  4  3  6

Enter Choice:
7
enter position
2
removed: 6
display:  4  3

Enter Choice:
```

PROGRAM 7

AIM : Program to implement stack operations using linked list.

INPUT

```
#include<stdio.h>
#include<stdlib.h>
struct node *l,*new,*ptr,*tm;
struct node
{
    int data;
    struct node *next;
};
struct node *head=NULL;
void create()
{
    int value;
    new=(struct node*)malloc(sizeof(struct node));
    printf("enter value to insert:\n");
    scanf("%d",&value);
    new->data=value;
    new->next=NULL;
}
void display()
{
    if(head==NULL)
    {
        printf("empty");
    }
    else
    {
        l=head;
```

```
            while(l!=NULL)
            {
                  printf(" %d ",l->data);
                  l=l->next;
            }
      }
      printf("\n\n");
}
void insert_end()
{
      create();
      if(head==NULL)
      {
            head=new;
      }
      else
      {
            l=head;
            while(l->next!=NULL)
            {
                  l=l->next;
            }
            l->next=new;
      }
      printf("display: ");
display();
}
void delete_end()
{
  if(head==NULL)
```

```
 {
  printf("empty");
 }
     else
     {        if(head->next==NULL)
              {
                       ptr=head;
                       head=NULL;
                       printf("removed: %d\n",ptr->data);
                       free(ptr);
              }
              else
              {
                       ptr=head;
                       while(ptr->next!=NULL)
                       {
                                tm=ptr;
                                ptr=ptr->next;
                       }
                       tm->next=NULL;
                       printf("removed: %d\n",ptr->data);
                       free(ptr);
              }
     }
printf("display: ");
display();
}
void main()
{    int choice;
              printf("1.insert at end (push)\n2.delete at end (pop)\n3.exit \n");
```

```
    while(1)
    {
        printf("1Enter Choice: \n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:insert_end();
            break;
            case 2:delete_end();
            break;
            case 3:exit(0);
            break;
            default: printf("wrong choice,%d is not valid\n",choice);
            break;
        }
    }
}
```

OUTPUT

```
stud@debian:~/elizabamca/ds$ gcc linli.c
stud@debian:~/elizabamca/ds$ ./a.out
1.insert at end (push)
2.delete at end(pop)
3.exit
Enter Choice:
1
enter value to insert:
3
display:  3

Enter Choice:
1
enter value to insert:
5
display:  3  5

Enter Choice:
1
enter value to insert:
7
display:  3  5  7

Enter Choice:
2
removed: 7
display:  3  5

Enter Choice:
2
removed: 5
display:  3

Enter Choice:
2
removed: 3
display: empty
```

PROGRAM 8

AIM : Program to implement queue operations using linked list.

INPUT

```
#include<stdio.h>
#include<stdlib.h>
struct node *l,*new,*ptr,*tm;
struct node
{
    int data;
    struct node *next;
};
struct node *head=NULL;
void create()
{
    int value;
    new=(struct node*)malloc(sizeof(struct node));
    printf("enter value to insert:\n");
    scanf("%d",&value);
    new->data=value;
    new->next=NULL;
}
void display()
{
    if(head==NULL)
    {
        printf("empty");
    }
    else
    {
        l=head;
```

```
            while(l!=NULL)
            {
                  printf(" %d ",l->data);
                  l=l->next;
            }
      }
      printf("\n\n");
}
void insert_end()
{
      create();
      if(head==NULL)
      {
            head=new;
      }
      else
      {
            l=head;
            while(l->next!=NULL)
            {
                  l=l->next;
            }
            l->next=new;
      }
      printf("display: ");
display();
}
void delete_beg()
{
  if(head==NULL)
```

```
 {
  printf("empty");
 }
     else
     {       if(head->next==NULL)
             {
                     ptr=head;
                     head=NULL;
                     printf("removed: %d\n",ptr->data);
                     free(ptr);
             }
             else
             {
                     ptr=head;
                     head=head->next;
                     printf("removed: %d\n",ptr->data);
                     free(ptr);
             }
     }
printf("display: ");
display();
}
void main()
{    int choice;
             printf("1.insert at end (enqueue)\n2.delete at beginning(dequeue)
\n3.exit \n");
     while(1)
     {
             printf("Enter Choice: \n");
             scanf("%d",&choice);
             switch(choice)
```

```
            {

                    case 1:insert_end();
                    break;
                    case 2:delete_beg();
                    break;
                    case 3:exit(0);
                    break;
                    default: printf("wrong choice,%d is not valid\n",choice);
                    break;
            }
      }
}
```

OUTPUT

```
stud@debian:~/elizabamca/ds$ gcc linli.c
stud@debian:~/elizabamca/ds$ ./a.out
1.insert at end (enqueue)
2.delete at beginning(dequeue)
3.exit
Enter Choice:
1
enter value to insert:
1
display:  1

Enter Choice:
1
enter value to insert:
3
display:  1  3

Enter Choice:
1
enter value to insert:
4
display:  1  3  4

Enter Choice:
1
enter value to insert:
5
display:  1  3  4  5

Enter Choice:
2
removed: 1
display:  3  4  5

Enter Choice:
2
removed: 3
display:  4  5
```

PROGRAM 9

AIM : Write a program to implement set operations

INPUT

```
#include<stdio.h>
#include<stdlib.h>
void input();
void setunion();
void intersection();
void compliment();
void display();
int n=5;
int a[5],b[5],c[5]; int main()
{int ch; while(1)
{printf("\nOPTIONS:\n");
printf("1.Input\n2.Union\n3.Intersection\n4.Compliment\n 5.Exit\n\n");
printf("Enter your choice\n");
scanf("%d",&ch); switch(ch)
{case 1:input();
break; case 2:setunion();
break;
case 3:intersection(); break;
case 4:compliment(); break;
case 5:exit;
break;
}
}
}
void input()
{
int n,x,i; printf("U={1,2,3,4,5}");
printf("\nEnter bitsring of set1\n"); for(i=0;i<5;i++)
```

```
{
scanf("%d",&a[i]);
}
printf("Enter bitstring of set2\n"); for(i=0;i<5;i++)
{
scanf("%d",&b[i]);
}
}
void display()
{
int i;
printf("Bitstring is \n"); for(i=0;i<n;i++)
{
printf("%d",c[i]);
}
printf("\n Set is\n"); for(i=0;i<n;i++)
{
if(c[i]!=0)
printf("%d",i+1);
}
}
void setunion()
{
int i;
printf("The union set of A and B is:\n"); for(i=0;i<n;i++)
{
if (a[i]!=b[i]) c[i]=1;
else c[i]=a[i];
}
display();
```

```
}
void intersection()
{
int i;
printf("The Intersection set of A and B is:\n"); for(i=0;i<n;i++)
{
if (a[i]!=b[i]) c[i]=0;
else c[i]=a[i];
}
display();
}
void compliment()
{
int i;
printf("\nThe compliment of set A:\n"); for(i=0;i<n;i++)
{
if (a[i]==1)
c[i]=0;
else c[i]=1;
}
display();
printf("\nThe compliment of set B:\n"); for(i=0;i<n;i++)
{
if (b[i]==1)
c[i]=0;
else c[i]=1;
}
display();
}
```

OUTPUT

```
OPTIONS:
1.Input
2.Union
3.Intersection
4.Compliment
 5.Exit

Enter  your choice
1
U={1,2,3,4,5}
Enter bitsring of set1
1 1 1 0 1
Enter bitstring of set2
0 0 1 1 1

OPTIONS:
1.Input
2.Union
3.Intersection
4.Compliment
 5.Exit

Enter  your choice
2
The union set of A and B is:
Bitstring is
11111
 Set is
12345
OPTIONS:
1.Input
2.Union
3.Intersection
4.Compliment
 5.Exit

Enter  your choice
3
The Intersection set of A and B is:
Bitstring is
00101
 Set is
35
OPTIONS:
1.Input
2.Union
3.Intersection
4.Compliment
 5.Exit

Enter  your choice
4

The compliment of set A:
Bitstring is
00010
 Set is
4
The compliment of set B:
Bitstring is
11000
 Set is
12
```

PROGRAM 10

AIM : Program to implement circular linked list

INPUT

```
#include<stdio.h>
#include<stdlib.h>
void create();
void insert_beg();
void insert_end();
void insert_pos();
void delete_beg();
void delete_end();
void delete_pos();
void display();
void insert_menu();
void delete_menu();
void menu_main();
struct node
{
int data;
struct node *next;
};struct node *new,*head;
struct node *l,*temp,*i,*del; void insert_beg()
{
int value;
new=(struct node *)malloc(sizeof(struct node)); printf("Enter the value to
insert:\n"); scanf("%d",&value);
new->data=value; new->next=NULL; if(head==NULL)
{
head=new;
new->next=head;
}
```

```
else
{
new->next=head; head=new;
}
}
void insert_end()
{
int value;
new=(struct node *)malloc(sizeof(struct node)); printf("Enter the value to insert:\n"); scanf("%d",&value);
new->data=value; new->next=NULL;
if(head==NULL)
{
head=new;
new->next=head;
}
else
{
l=head;
while(l->next!=head)
{
l=l->next;
}
l->next=new; new->next=head;
}
}
void insert_pos()
{
int value,pos;
new=(struct node *)malloc(sizeof(struct node));
printf("Enter position:\n"); scanf("%d",&pos);
```

```
printf("Enter the value to insert:\n",pos); scanf("%d",&value);
new->data=value; new->next=NULL; if(head==NULL)
{
head=new;
new->next=head;
}
else
{
if(pos==0)
{
new->next=head; head=new;
}
else
{
l=head; int i;
for(i=0;i<pos-1;i++)
{
l=l->next; if(l==NULL)
{
printf("linked list size is less than the given position\n");
}
}
new->next=l->next; l->next=new;
}
}
}
void display()
{
if(head==NULL) printf("List is empty\n"); else
{
```

```
l=head;
printf("\nThe elements are:\n"); do
{
printf("%d\n",l->data); l=l->next;
} while(l!=head);
}
} void delete_beg()
{
if(head==NULL) printf("List is empty\n"); else
{
if(head->next==head)
{
temp=head; head=NULL;
printf("\nRemoved %d\n",temp->data); free(temp);
}
else
{
temp=head; head=temp->next;
printf("Removed:%d\n",temp->data); free(temp);
}
}
}
void delete_end()
{
if(head==NULL) printf("List is empty\n"); else
{
if(head->next==head)
{
temp=head; head=NULL;
printf("\nRemoved %d\n",temp->data); free(temp);
```

```
}
else
{
temp=head;
while(temp->next!=head)
{
del=temp; temp=temp->next;
}
del->next=head;
printf("\nRemoved %d\n",temp->data); free(temp);
}
}
}
void delete_pos()
{
int pos; if(head==NULL)
{
printf("List is empty\n");
}
else
{
l=head;
printf("Enter the position:\n"); scanf("%d",&pos);
if(pos==0)
{
delete_beg();
}
else
{
del=head; int i;
```

```
for(i=0;i<pos;i++)
{
temp=del; del=del->next; if(del==NULL)
{
printf("Error");
}
}
temp->next=del->next; printf("Removed %d\n",del->data); free(del);
}
}
}
void insert_menu()
{
int ch;
while(1)
{
printf("1.Insert at beginning\n2.Insert at end\n3.Insert at a particular
position\n4.Display linked list\n5.Exit\n");
printf("Enter your choice:\n"); scanf("%d",&ch);
switch(ch)
{
case 1: insert_beg()
break;
case 2: insert_end(); break;
case 3: insert_pos(); break;
case 4: display();
break;
case 5: menu_main(); break;
default: printf("Invalid Choice\n");
}
}
```

```
}
void delete_menu()
{
int ch; while(1)
{
printf("1.Delete at beginning\n2.Delete at end\n3.Delete from a particular position\n4.Display linked list\n5.Exit\n");
printf("Enter your choice:\n"); scanf("%d",&ch);
switch(ch)
{
case 1: delete_beg(); break;
case 2: delete_end(); break;
case 3: delete_pos(); break;
case 4: display();
break;
case 5: menu_main()
default: printf("Invalid Choice\n");
}
}
}
void menu_main()
{
int ch; while(1)
{
printf("1.Insert into linked list\n2.Delete form linked list\n3.Exit\n"); printf("Enter your choice:\n");
scanf("%d",&ch);{ switch(ch)
{
case 1: insert_menu(); break;
case 2: delete_menu();
break; case 3: exit(1);
```

```
break;
default: printf("Invalid Choice\n");
}
}
}
}
void main()
{
printf("CIRCULAR LINKED LIST OPERATIONS\n"); printf("          \n\n");
menu_main();
}
```

OUTPUT

```
CIRCULAR LINKED LIST OPERATIONS
................................

1.Insert into linked list
2.Delete form linked list
3.Exit
Enter your choice:
1
1.Insert at beginning
2.Insert at end
3.Insert at a particular position
4.Display linked list
5.Exit
Enter your choice:
1
Enter the value to insert:
6
1.Insert at beginning
2.Insert at end
3.Insert at a particular position
4.Display linked list
5.Exit
Enter your choice:
2
Enter the value to insert:
7
1.Insert at beginning
2.Insert at end
3.Insert at a particular position
4.Display linked list
5.Exit
Enter your choice:
3
Enter position:
1
Enter the value to insert:
4
```

```
CIRCULAR LINKED LIST OPERATIONS
....................................

1.Insert into linked list
2.Delete form linked list
3.Exit
Enter your choice:
2
1.Delete at beginning
2.Delete at end
3.Delete from a particular position
4.Display linked list
5.Exit
Enter your choice:
1
List is empty
```

PROGRAM 11

AIM : Program to implement doubly linked list

INPUT

```
#include<stdio.h>
#include<stdlib.h>
struct node *l,*new,*ptr,*tm;
struct node
{
    struct node *prev;
    int data;
    struct node *next;
};
struct node *head=NULL;
void create()
{
    int value;
    new=(struct node*)malloc(sizeof(struct node));
    printf("enter value to insert:\n");
    scanf("%d",&value);
    new->data=value;
    new->next=NULL;
    new->prev=NULL;
}
void display()
{
    if(head==NULL)
    {
        printf("empty");
    }
    else
```

```
        {
                l=head;
                while(l!=NULL)
                {
                        printf("%d ",l->data);
                        l=l->next;
                }
        }
        printf("\n\n");
}
void insert_beg()
{       create();
        if(head==NULL)
        {
                head=new;
        }
        else
        {
                new->next=head;
                head->prev=new;
                head=new;
        }
        printf("display: ");
display();
}
void insert_end()
{
        create();
        if(head==NULL)
        {
```

```
            head=new;
        }
        else
        {
            l=head;
            while(l->next!=NULL)
            {
                l=l->next;
            }
            new->prev=l;
            l->next=new;
        }
        printf("display: ");
display();
}
void insert_pos()
{
        create();
        if(head==NULL)
        {
            printf("empty");
            head=new;
        }
        else
        {
            l=head;
            int pos,i;
            printf("enter position to insert:\n");
            scanf("%d",&pos);
            if(pos==0)
```

```
                {
                insert_beg();
                }
        else
                for(i=0;i<pos-1;i++)
                {
                        l=l->next;
                }
                new->prev=l;
                new->next=l->next;
                l->next->prev=new;
                l->next=new;
    }
    printf("display: ");
    display();
}
void delete_beg()
{
 if(head==NULL)
 {
  printf("empty");
 }
    else
    {       if(head->next==NULL)
            {
                ptr=head;
                head=NULL;
                printf("removed: %d\n",ptr->data);
                free(ptr);
            }
```

```
            else
            {
                  ptr=head;
                  head=head->next;
                  head->prev=NULL;
                  printf("removed: %d\n",ptr->data);
                  free(ptr);
            }
      }
printf("display: ");
display();
}
void delete_end()
{
 if(head==NULL)
 {
  printf("empty");
 }
     else
     {      if(head->next==NULL)
            {
                  ptr=head;
                  head=NULL;
                  printf("removed: %d\n",ptr->data);
                  free(ptr);
            }
            else
            {
                  ptr=head;
                  while(ptr->next!=NULL)
```

```
                {
                        ptr=ptr->next;
                }
                printf("removed: %d\n",ptr->data);
                ptr->prev->next=NULL;
                free(ptr);
        }
    }
printf("display: ");
display();
}
void delete_pos()
{
    if(head==NULL)
    {
    printf("empty");
    }
    else
    {
    int pos;
    printf("enter position \n");
    scanf("%d",&pos);
        if(pos==0)
        {
                delete_beg();
        }
        else
        {       ptr=head;
                for(int i=0;i<pos-1;i++)
                {
```

```
                        ptr=ptr->next;
                }
                printf("removed: %d\n",ptr->data);
                ptr->prev->next=ptr->next;
                ptr->next->prev=ptr->prev;
                free(ptr);
        }
}
printf("display: ");
display();
}
void main()
{
    while(1)
    {       int choice;
            printf("1.insert at beggining \n2.insert at end \n3.insert at position
\n4.delete at beginning \n5.delete at end \n6.display  \n7.delete at position \n8.exit
\nEnter Choice: \n");
            scanf("%d",&choice);
            switch(choice)
            {
                    case 1:insert_beg();
                    break;
                    case 2:insert_end();
                    break;
                    case 3:insert_pos();
                    break;
                    case 4:delete_beg();
                    break;
                    case 5:delete_end();
                    break;
```

```
                    case 6:printf("status: ");
                           display();
                    break;
                    case 7:delete_pos();
                    break;
                    case 8:exit(0);
                    break;
                    default: printf("wrong choice,%d is not valid\n",choice);
                    break;
            }
      }
}
```

OUTPUT

```
stud@debian:~/elizabamca/ds$ gcc linli.c
stud@debian:~/elizabamca/ds$ ./a.out
1.insert at beggining
2.insert at end
3.insert at position
4.delete at beginning
5.delete at end
6.display
7.delete at position
8.exit
Enter Choice:
1
enter value to insert:
3
display:  3

Enter Choice:
1
enter value to insert:
4
display:  4  3

Enter Choice:
2
enter value to insert:
5
display:  4  3  5

Enter Choice:
3
enter value to insert:
6
enter position to insert:
2
display:  4  3  6  5


Enter Choice:
1
enter value to insert:
7
display:  7  4  3  6  5

Enter Choice:
4
removed: 7
display:  4  3  6  5

Enter Choice:
5
removed: 5
display:  4  3  6

Enter Choice:
7
enter position
2
removed: 6
display:  4  3

Enter Choice:
```

PROGRAM 12

AIM : Write a program to implement binary search tree.

INPUT

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

struct node
{
int data;
struct node *lchild; struct node *rchild;
};
struct node *root=NULL,*new,*p,*ptr,*t,*tp; int value;

void delet(struct node *ptr,struct node *p)
{
if((ptr->lchild==NULL)&&(ptr->rchild==NULL))
{
if(ptr==p)
{
root=NULL;
}
else if(p->lchild==ptr)
{
p->lchild=NULL;
}
else if(p->rchild==ptr)
{
p->rchild=NULL;
}
else
{
printf("Invalid Choice");
}
printf("Removed %d\n",ptr->data); free(ptr);
}
else if((ptr->lchild!=NULL)&&(ptr->rchild==NULL))
{
if(ptr==p)
{
root=ptr->lchild;
}

else if(p->lchild==ptr)
{
p->lchild=ptr->lchild;
}
else if(p->rchild==ptr)
{
```

```
p->rchild=ptr->lchild;
}
else
{
printf("Invalid Choice");
}
printf("Removed %d\n",ptr->data); free(ptr);
}
else if((ptr->lchild==NULL)&&(ptr->rchild!=NULL))
{
if(ptr==p)
{
root=ptr->rchild;
}
else if(ptr==p->lchild)
{
p->lchild=ptr->rchild;
}
else if(p->rchild==ptr)
{
p->rchild=ptr->rchild;
}
else
{
printf("Invalid choice");
}
printf("Removed %d\n",ptr->data); free(ptr);
}
else if((ptr->rchild!=NULL)&&(ptr->lchild!=NULL))
{
t=ptr->rchild;
while(t->lchild!=NULL)
{
tp=t;
t=t->lchild;
}
ptr->data=t->data; tp->lchild=NULL;
printf("Removed %d\n",ptr->data); free(ptr);
}
}
void search2(struct node *rt,int val)
{
if(val>rt->data)
{
p=rt;
search2(rt->rchild,value);
}
else if(val<rt->data)
{
p=rt;
```

```
search2(rt->lchild,value);
}
else if(val==rt->data)
{
delet(rt,p);
}
 else
}
{
printf("Node doesn't exist\n");
}
void delete()
{
if(root==NULL)
{
printf("Tree is empty\n");
}
else
}
{
printf("Enter the value to remove\n"); scanf("%d",&value); search2(root,value);
}
void display(struct node *rt)
{
if(rt!=NULL)
{
printf(" %d\t",rt->data); display(rt->rchild); display(rt->lchild);
}
printf("\n");
}
void search(struct node *rt)
{
if((new->data>rt->data)&&(rt->rchild==NULL))
{
rt->rchild=new; printf("Inserted\n");
}
else if((new->data>rt->data)&&(rt->rchild!=NULL))
{
search(rt->rchild);
}
else if((new->data<rt->data)&&(rt->lchild==NULL))
{
rt->lchild=new; printf("Inserted\n");
}
else if((new->data<rt->data)&&(rt->lchild!=NULL))
{
search(rt->lchild);
}
 else
}
```

```
{
printf("Invalid Choice\n");
}
void insert()
{
new=(struct node *)malloc(sizeof(struct node)); printf("Enter the value to
insert\n"); scanf("%d",&value);
new->data=value; new->lchild=NULL; new->rchild=NULL;
if(root==NULL)
{
root=new; printf("Inserted\n");
}
else
{
search(root);
}
}
void main()
{
int choice; while(1)
{
printf("1.Insertion\n2.Deletion\n3.Display\n4.Exit\n"); printf("Enter your
choice:");
scanf("%d",&choice); switch(choice)
{
case 1:insert();
break; case 2:delete();
break;
case 3:display(root); break;
case 4:exit(1);
break; default:printf("Invalid choice\n");
break;
}
}
}
```

OUTPUT

```
1.Insertion
2.Deletion
3.Display
4.Exit
Enter your choice:1
Enter the value to insert
12
Inserted
1.Insertion
2.Deletion
3.Display
4.Exit
Enter your choice:1
Enter the value to insert
10
Inserted
1.Insertion
2.Deletion
3.Display
4.Exit
Enter your choice:1
Enter the value to insert
19
Inserted
1.Insertion
2.Deletion
3.Display
4.Exit
Enter your choice:3
  12      19


  10

.
  4.Exit
  Enter your choice:1
  Enter the value to insert
  20
  Inserted
  1.Insertion
  2.Deletion
  3.Display
  4.Exit
  Enter your choice:3
    12      19      20




    10



  1.Insertion
  2.Deletion
  3.Display
  4.Exit
  Enter your choice:2
  Enter the value to remove
  20
  Removed 20
  1.Insertion
  2.Deletion
  3.Display
  4.Exit
  Enter your choice:3
    12      19


    10
```

PROGRAM 13

AIM : Write a program for Breadth First Search(BFS)
INPUT

```
#include<stdio.h>
#include<stdlib.h>
int q[20],front=-1,rear=-1,arr[20][20],visited[20]={0};
 int bfs (int s, int n);
void main()
{int i,j,n,ch,s;
printf("       BFS     \n");
printf("Enter the Number of Vertices\n");
scanf("%d",&n);
printf("Enter the adjacency matrix:\n");
for(i=1;i<=n;i++)
{for(j=1;j<=n;j++)
{scanf("%d",&arr[i][j]);
}}
printf("Enter stating vertex "); scanf("%d",&s);
bfs(s,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);
}
}
int bfs(int s,int n)
{int i,p; add(s); visited[s]=1; p=delete(); if(p!=0)
printf("%d\t",p); while(p!=0)
{
for(i=1;i<=n;i++)
{
if((arr[p][i]!=0)&&(visited[i]==0))
{add(i); visited[i]=1;
}
}
p=delete(); if(p!=0) printf("%d\t",p);}}
```

OUTPUT

```
-------------BFS---------------
Enter the Number of Vertices
4
Enter the adjacency matrix:
1 1 0 1
0 0 1 1
1 0 1 0
1 1 1 1
Enter stating vertex 4
4       1       2       3
[Process completed (code 127) - press Enter]
```

PROGRAM 14

AIM : Write a program for Depth First Search(DFS)
INPUT

```
#include<stdio.h>
#include<stdlib.h>
int top=-1,stack[20],arr[20][20],visited[20]={0};
void dfs (int s, int n);
void main()
{
int i,j,n,ch,s;
printf("       DFS     \n");
printf("Enter the Number of Vertices\n");
scanf("%d",&n);
printf("Enter the adjacency matrix:\n");
for(i=1;i<=n;i++)
{for(j=1;j<=n;j++)
{scanf("%d",&arr[i][j]);
}
}
printf("Enter stating vertex "); scanf("%d",&s);
dfs(s,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 );
}
}void dfs(int s,int n)
{
int k,i; push(s); visited[s]=1;
k=pop(); if(k!=0)
printf("%d\t",k); while(k!=0)
{
for(i=1;i<=n;i++)
{
if((arr[k][i]!=0)&&(visited[i]==0))
{
push(i); visited[i]=1;
}
}
k=pop();
if(k!=0) printf("%d\t",k);
}
}
```

OUTPUT

```
------------DFS-------------
Enter the Number of Vertices
4
Enter the adjacency matrix:
1 1 0 1
0 1 1 0
1 1 1 1
0 0 1 0
Enter stating vertex 3
3       4       2       1
[Process completed - press Enter]
```

.

PROGRAM 15

AIM : Write a program to implement Prim's algorithm.
INPUT

```
#include<stdio.h>
#define INF 9999999
int V,i,j;
int G[10][10];
int visited[10]={0};
int main()
{
printf("Enter the number of vertices:"); scanf("%d",&V);
printf("Enter the cost adjacency matrix : \n"); for(i=1;i<=V;i++)
{
for(j=1;j<=V;j++)
{
scanf("%d",&G[i][j]); if(G[i][j]==0)
{ G[i][j]=INF;
}
}
}
int no_edge=0; // number of edge visited[1] = 1;
int x,y; // row,col number int min_cost=0;
printf("\n The edges of spanning tree are:\nEdge : Weight\n");
while (no_edge<V-1)
{
int min = INF; x = 0;
y = 0;
for (int i = 1; i <= V; i++)
{
if (visited[i]==1)
{
for (int j = 1; j <= V; j++)
{
if (visited[j]==0 && G[i][j]!=INF)
{ // not visited and there is an edge if (G[i][j]<min)
{
min = G[i][j]; x = i;
y = j;
}
}
}
}
}
printf("%d - %d : %d\n", x, y, G[x][y]); visited[y]=1;
 no_edge++; min_cost=min_cost+min;
}
printf("total cost:%d",min_cost); return 0;
}
```

OUTPUT

```
Enter the number of vertices:4
Enter the cost adjacency matrix :
0 4 10 6 3
4 0 9 0 0
10 9 0 0 1
6 0 0 0 7

 The edges of spanning tree are:
Edge : Weight
1 - 2 : 4
1 - 4 : 6
4 - 3 : 1
total cost:11
[Process completed - press Enter]
```

PROGRAM 16

AIM : Write a program to implement Kruskal's algorithm..

INPUT

```
#include<stdio.h>
#include<stdlib.h>
#define INF 1000
int edge=0;
int i,j,cost[20][20],n,visited[20]={0},a,b,mincost=0;
void main()
{
printf("Enter the number of vertices:"); scanf("%d",&n);
printf("Enter 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]=INF;
}
}
}
for(int k=1;k<n;k++)
{
int min=INF; for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
if(cost[i][j]<min)
{
min=cost[i][j]; a=i;
b=j;
}
}
}


printf("\nEdge :%d %d-->%d\n",k,a,b); printf("\nCost :%d\n",min);
mincost=mincost+min;
visited[b]=1;


cost[a][b]=cost[b][a]=INF;
}
printf("\nTotal cost cost is :%d",mincost);
}
```

OUTPUT

```
Enter the number of vertices:5
Enter the cost adjacency matrix :
0 4 10 6 3
4 0 9 0 0
10 9 0 0 1
6 0 0 0 7
3 0 1 7 0

Edge :1 3-->5

Cost :1

Edge :2 1-->5

Cost :3

Edge :3 1-->2

Cost :4

Edge :4 1-->4

Cost :6
```

Total cost is :14