VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
On

DATA STRUCTURES (23CS3PCDST)

Submitted by

AKSHAY S (1BM23CS022)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
September 2024-January 2025

B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering

This is to certify that the Lab work entitled **"DATA STRUCTURES"** carried out by AKSHAY S **(1BM23CS022)**, who is bonafide student of **B. M. S. College of Engineering**. It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of Data structures Lab - **(23CS3PCDST)**work prescribed for the said degree.

Prof. Rajeshwari B S
Associate Professor
Department of CSE
BMSCE, Bengaluru

Dr. Kavitha Sooda
Professor and Head
Department of CSE
BMSCE, Bengaluru

Index Sheet

Course outcomes:

CO1	Apply the concept of linear and nonlinear data structures.
CO2	Analyze data structure operations for a given problem
CO3	Design and develop solutions using the operations of linear and nonlinear data structure for a given specification.
CO4	Conduct practical experiments for demonstrating the operations of different data structures.

Lab program 1:

Write a program to simulate the working of stack using an array with the following: a) Push b) Pop c) Display. The program should print appropriate messages for stack overflow, stack underflow.

Code:

```
#include<stdio.h>
#include<conio.h>
#define max 3
void push();
int pop();
void display();
int s[10],item,top=-1,i,ch;
void main()
{
   while(1)
   {
      printf(" 1:Push\n 2:Pop\n 3:Display\n 4:Exit\n");
      printf("Enter your choice:");
      scanf("%d",&ch);
      switch(ch)
      {
         case 1:push();
               break;
         case 2:item=pop();
               if(item!=-1)
                printf("Popped element is %d\n",item);
                break;
         case 3:display();
               break;
         case 4:exit(0);
               break;
      }
   }
   getch();
}
void push()
{
   if(top==max-1)
   {
      printf("STACK OVERFLOW\n");
      return;
   }
   printf("Enter element to be pushed:");
```

```
   scanf("%d",&item);
   top=top+1;
   s[top]=item;
}
int pop()
{
   if(top==-1)
   {
      printf("STACK UNDERFLOW\n");
      return(-1);
   }
   item=s[top];
   top=top-1;
   return item;
}
void display()
{
   if(top==-1)
   {
      printf("Stack is empty\n");
      return;
   }
   printf("Stack contents:\n");
   for(i=top;i>=0;i--)
   {
      printf("%d\n",s[i]);
   }
   return;
}
```

Output:

```
 1:Push
 2:Pop
 3:Display
 4:Exit
Enter your choice:1
Enter element to be pushed:10
 1:Push
 2:Pop
 3:Display
 4:Exit
Enter your choice:1
Enter element to be pushed:20
 1:Push
 2:Pop
 3:Display
 4:Exit
Enter your choice:1
Enter element to be pushed:30
 1:Push
 2:Pop
 3:Display
 4:Exit
Enter your choice:1
STACK OVERFLOW
 1:Push
 2:Pop
 3:Display
 4:Exit
Enter your choice:2
Popped element is 30
 1:Push
 2:Pop
 3:Display
 4:Exit
Enter your choice:3
Stack contents:
20
10
 1:Push
 2:Pop
 3:Display
 4:Exit
Enter your choice:2
Popped element is 20
```

```
 1:Push
 2:Pop
 3:Display
 4:Exit
Enter your choice:3
Stack contents:
10
 1:Push
 2:Pop
 3:Display
 4:Exit
Enter your choice:2
Popped element is 10
 1:Push
 2:Pop
 3:Display
 4:Exit
Enter your choice:3
Stack is empty
 1:Push
 2:Pop
 3:Display
 4:Exit
Enter your choice:4
```

Lab program 2:

Write a program to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide).

Code:

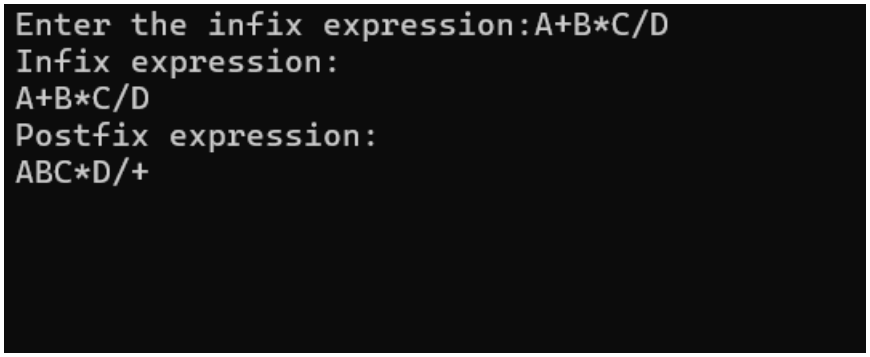
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int top=-1,index=0,pos=0,len;
char symbol,temp,infix[20],stack[20],postfix[20];
void push(char symbol);
char pop();
char pred(char symbol);
void infixtopostfix();
void main()
{
   printf("Enter the infix expression:");
   scanf("%s",infix);
   infixtopostfix();
   printf("Infix expression:\n%s",infix);
   printf("\nPostfix expression:\n%s",postfix);
   getch();
}
void infixtopostfix()
{
   len=strlen(infix);
   push('#');
   while(index<len)
   {
      symbol=infix[index];
```

```
    switch(symbol)
    {
       case '(':push(symbol);
              break;
       case ')':temp=pop();
              while(temp!='(')
              {
                 postfix[pos]=temp;
                 pos++;
                 temp=pop();
              }
              break;
       case '+':
       case '-':
       case '*':
       case '/':
       case '^':while(pred(stack[top])>=pred(symbol))
              {
                 temp=pop();
                 postfix[pos++]=temp;
              }
              push(symbol);
              break;
       default:postfix[pos++]=symbol;
    }
    index++;
 }
 while(top>0)
 {
```

```
        temp=pop();
        postfix[pos++]=temp;
    }
}
void push(char symbol)
{
    top=top+1;
    stack[top]=symbol;
}
char pop()
{
    char symbol;
    symbol=stack[top];
    top=top-1;
    return(symbol);
}
char pred(char symbol)
{
    int p;
    switch(symbol)
    {
        case '^':p=100;
                 break;
        case '*':
        case '/':p=80;
                 break;
        case '+':
        case '-':p=60;
                 break;
```

```
        case '(':p=40;
                break;
        case '#':p=20;
                break;
    }
    return(p);
}
```

Output:



Lab program 3:

Queue implementation

Code:

```
#include<stdio.h>
#define max 3
int q[20],front=-1,rear=-1,ch,ele,i;
void insert();
int delete();
void display();
void main()
{
   while(1)
   {
      printf("\n1.Insert\n2.Delete\n3.Display\n4.Exit\n");
      printf("Enter your choice:");
      scanf("%d",&ch);
      switch(ch)
      {
         case 1:insert();
                break;
         case 2:ele=delete();
                if(ele!=-1)
                {
                   printf("Deleted element is %d",ele);
                }
                break;
         case 3:display();
                break;
         case 4:exit(0);
```

```
                break;
            default :printf("Wrong choice");
        }
    }
}
void insert()
{
    if(rear==max-1)
    {
        printf("Queue if full");
        return;
    }
    if(rear==-1)
    {
        rear=0;
        front=0;
    }
    else
    {
        rear=rear+1;
    }
    printf("Enter element to be inserted:");
    scanf("%d",&ele);
    q[rear]=ele;
}
int delete()
{
    if(front==-1)
    {
```

```
        printf("Queue is empty");
        return(-1);
    }
    ele=q[front];
    if(front==rear)
    {
        front=-1;
        rear=-1;
    }
    else
    {
        front=front+1;
    }
    return(ele);
}
void display()
{
    if(front==-1)
    {
        printf("Queue is empty");
        return;
    }
    printf("Queue contents:\n");
    for(i=front;i<=rear;i++)
    {
        printf("%d\t",q[i]);
    }
    return;
}
```

Output:

```
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:1
Enter element to be inserted:10

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:1
Enter element to be inserted:20

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:1
Enter element to be inserted:30

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:1
Queue if full
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:3
Queue contents:
10      20      30
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:2
Deleted element is 10
1.Insert
2.Delete
3.Display
4.Exit
```

```
Enter your choice:3
Queue contents:
20      30
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:2
Deleted element is 20
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:3
Queue contents:
30
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:2
Deleted element is 30
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:3
Queue is empty
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:4
```

Lab program 4:

Circular Queue implementation

Code:

```
#include<stdio.h>
#define size 4
int cq[20],i,ch,front=-1,rear=-1,item;
void insert();
int delete();
void display();
void main()
{
   while(1)
   {
      printf("\n1.Insert\n2.Delete\n3.Display\n4.Exit\n");
      printf("Enter your choice:");
      scanf("%d",&ch);
      switch(ch)
      {
         case 1:insert();
                break;
         case 2:item=delete();
                if(item!=-1)
                {
                   printf("Deleted element is %d",item);
                }
                break;
         case 3:display();
                break;
         case 4:exit(0);
```

```
                break;
        }
    }
}
void insert()
{
    if(front==(rear+1)%size)
    {
        printf("Circular queue is full");
        return;
    }
    if(front==-1 && rear==-1)
    {
        front=0;
        rear=0;
    }
    else
    {
        rear=(rear+1)%size;
    }
    printf("Enter element to be inserted:");
    scanf("%d",&item);
    cq[rear]=item;
    return;
}
int delete()
{
    if(front==-1 && rear==-1)
    {
```

```
        printf("Circular queue is empty");
        return(-1);
    }
    item=cq[front];
    if(front==rear)
    {
        front=-1;
        rear=-1;
    }
    else
    {
        front=(front+1)%size;
    }
    return(item);
}
void display()
{
    if(front==-1 && rear==-1)
    {
        printf("Circular queue is empty");
        return;
    }
    printf("Circular queue contains:\n");
    if(front<=rear)
    {
        for(i=front;i<=rear;i++)
        {
            printf("%d\t",cq[i]);
        }
```

```
    }
    else
    {
        for(i=front;i<=size-1;i++)
        {
            printf("%d\t",cq[i]);
        }
        for(i=0;i<=rear;i++)
        {
            printf("%d\t",cq[i]);
        }
    }
    return;
}
```

Output:

```
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:1
Enter element to be inserted:10

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:1
Enter element to be inserted:20

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:1
Enter element to be inserted:30

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:1
Enter element to be inserted:40

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:1
Circular queue is full
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:3
Circular queue contains:
10      20      30      40
1.Insert
2.Delete
3.Display
4.Exit
```

```
Enter your choice:2
Deleted element is 10
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:3
Circular queue contains:
20      30      40
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:1
Enter element to be inserted:50

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:3
Circular queue contains:
20      30      40      50
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:2
Deleted element is 20
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:2
Deleted element is 30
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:2
Deleted element is 40
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:2
```

```
Deleted element is 50
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:2
Circular queue is empty
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:3
Circular queue is empty
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:4
```

Lab program 5:

1) **Factorial using recursion**
2) **Fibonacci using recursion**
3) **Tower of Hanoi using recursion**

Code:

1) **Factorial using recursion**

```
#include<stdio.h>
int fact(int n)
{
   int f;
   if(n==0 || n==1)
   {
      f=1;
   }
   else
   {
      f=n*fact(n-1);
   }
   return f;
}
void main()
{
   int n,t,f1;
   printf("Enter number:");
   scanf("%d",&n);
   t=n;
   f1=fact(n);
   printf("Factorial of %d is %d",t,f1);
```

```
}
```

2) Fibonacci using recursion

```
#include<stdio.h>
int fibo(int n)
{
   if(n==1)
   {
      return 0;
   }
   else if(n==2)
   {
      return 1;
   }
   else
   {
      return fibo(n-1)+fibo(n-2);
   }
}
void main()
{
   int n,fib;
   printf("Enter number:");
   scanf("%d",&n);
   fib=fibo(n);
   printf("Fibonacci number:%d",fib);
}
```

3) Tower of Hanoi using recursion

```
#include<stdio.h>
void toh(int n,char s,char t,char d)
{
   if(n==1)
   {
      printf("Move %d from %c to %c\n",n,s,d);
   }
   else
   {
      toh(n-1,s,d,t);
      printf("Move %d from %c to %c\n",n,s,d);
      toh(n-1,t,s,d);

   }
}
void main()
{
   int n;
   char s='S',d='D',t='T';
   printf("Enter number:");
   scanf("%d",&n);
   toh(n,s,t,d);
}
```

Output:

```
Enter number:5
Factorial of 5 is 120
Process returned 21 (0x15)   execution time : 2.582 s
Press any key to continue.
```

```
Enter number:8
Fibonacci number:13
Process returned 19 (0x13)   execution time : 3.011 s
Press any key to continue.
```

```
Enter number:3
Move 1 from S to D
Move 2 from S to T
Move 1 from D to T
Move 3 from S to D
Move 1 from T to S
Move 2 from T to D
Move 1 from S to D
```

Lab program 6:

1) **WAP to Implement Singly Linked List with following operations**
   a) Create a linked list.
   b) Insertion of a node at first position, at any position and at end of list.
   c) Display the contents of the linked list.
2) **WAP to Implement Singly Linked List with following operations**
   a) Create a linked list.
   b) Deletion of first element, specified element and last element in the list.
   c) Display the contents of the linked list.

Code:

1)

```
#include<stdio.h>
struct Node{
   int data;
   struct Node *link;
};
typedef struct Node node;
node *start=NULL,*curr,*temp,*new1;
void create();
void display();
void insert_beg();
void insert_end();
void insert_at_given_positon();
int ch,pos;
char c;
void main(){
   while(1){
      printf("\n1.Create\n2.Insert Beginnig\n3.Insert End\n4.Insert at given position\n5.Display\n6.Exit\n");
      printf("Enter your choice:");
      scanf("%d",&ch);
      switch(ch){
```

```
            case 1:create();
                    break;
            case 2:insert_beg();
                    break;
            case 3:insert_end();
                    break;
            case 4:insert_at_given_position();
                    break;
            case 5:display();
                    break;
            case 6:exit(0);
                    break;
        }
    }
}
void create(){
    do{
        new1=(node*)malloc(sizeof(node));
        printf("Enter element:");
        scanf("%d",&new1->data);
        if(start==NULL){
            start=new1;
            curr=new1;
        }else{
            curr->link=new1;
            curr=new1;
        }
        printf("If you want to add another element (Y/N):");
        scanf("%s",&c);
```

```
    }while(c=='Y' || c=='y');
    curr->link=NULL;
}
void display(){
    if(start==NULL){
        printf("Linked list is empty");
        return;
    }
    printf("Elements in linked list are:\n");
    temp=start;
    while(temp!=NULL){
        printf("%d\t",temp->data);
        temp=temp->link;
    }
}
void insert_beg(){
    new1=(node*)malloc(sizeof(node));
    printf("Enter element:");
    scanf("%d",&new1->data);
    if(start==NULL){
        start=new1;
        new1->link=NULL;
        return;
    }
    new1->link=start;
    start=new1;
}
void insert_end(){
    new1=(node*)malloc(sizeof(node));
```

```
    printf("Enter element:");
    scanf("%d",&new1->data);
    if(start==NULL){
        start=new1;
        new1->link=NULL;
        return;
    }
    temp=start;
    while((temp->link)!=NULL){
        temp=temp->link;
    }
    temp->link=new1;
    new1->link=NULL;
}
void insert_at_given_position(){
    new1=(node*)malloc(sizeof(node));
    printf("Enter element:");
    scanf("%d",&new1->data);
    printf("Enter position:");
    scanf("%d",&pos);
    if(pos==1){
        new1->link=start;
        start=new1;
        return;
    }
    temp=start;
    int i=1;
    while(temp!=NULL && i<pos-1){
        temp=temp->link;
```

```
        i++;
    }
    if(temp==NULL){
        printf("Entered position is greater than number of elements");
        return;
    }
    new1->link=temp->link;
    temp->link=new1;
}
```

2)

```
#include<stdio.h>
struct Node{
    int data;
    struct Node *link;
};
typedef struct Node node;
node *start=NULL,*curr,*temp,*new1,*pre,*next;
void create();
void display();
void delete_first();
void delete_last();
void delete_specific_element();
int ch,pos,ele;
char c;
void main(){
    while(1){
        printf("\n1.Create\n2.Delete First element\n3.Delete last element\n4.Delete specific element\n5.Display\n6.Exit\n");
        printf("Enter your choice:");
        scanf("%d",&ch);
```

```
        switch(ch){
            case 1:create();
                   break;
            case 2:delete_first();
                   break;
            case 3:delete_last();
                   break;
            case 4:delete_specific_element();
                   break;
            case 5:display();
                   break;
            case 6:exit(0);
                   break;
        }
    }
}
void create(){
    do{
        new1=(node*)malloc(sizeof(node));
        printf("Enter element:");
        scanf("%d",&new1->data);
        if(start==NULL){
            start=new1;
            curr=new1;
        }else{
            curr->link=new1;
            curr=new1;
        }
        printf("If you want to add another element (Y/N):");
```

```
        scanf("%s",&c);
    }while(c=='Y' || c=='y');
    curr->link=NULL;
}
void display(){
    if(start==NULL){
        printf("Linked list is empty");
        return;
    }
    printf("Elements in linked list are:\n");
    temp=start;
    while(temp!=NULL){
        printf("%d\t",temp->data);
        temp=temp->link;
    }
}
void delete_first(){
    if(start==NULL){
        printf("Linked list is empty");
        return;
    }
    temp=start;
    start=start->link;
    free(temp);
}
void delete_last(){
    if(start==NULL){
        printf("Linked list is empty");
        return;
```

```
    }
    if(start->link==NULL){
        temp=start;
        start=NULL;
        free(temp);
        return;
    }
    pre=NULL;
    next=start;
    while(next->link!=NULL){
        pre=next;
        next=next->link;
    }
    pre->link=NULL;
    free(next);
}
void delete_specific_element(){
    printf("Enter element to be deleted:");
    scanf("%d",&ele);
    if(start==NULL){
        printf("Linked list is empty");
        return;
    }
    if(start->data==ele){
        temp=start;
        start=start->link;
        free(temp);
        return;
    }
```

```
    pre=NULL;
    next=start;
    while(next->data!=ele && next->link!=NULL){
        pre=next;
        next=next->link;
    }
    if(next->data==ele){
        pre->link=next->link;
        free(next);
        return;
    }
    else{
        printf("Element not found");
    }
}
```

Output:

```
1.Create
2.Insert Beginnig
3.Insert End
4.Insert at given position
5.Display
6.Exit
Enter your choice:1
Enter element:10
If you want to add another element (Y/N):y
Enter element:20
If you want to add another element (Y/N):n

1.Create
2.Insert Beginnig
3.Insert End
4.Insert at given position
5.Display
6.Exit
Enter your choice:5
Elements in linked list are:
10      20
1.Create
2.Insert Beginnig
3.Insert End
4.Insert at given position
5.Display
6.Exit
Enter your choice:2
Enter element:0

1.Create
2.Insert Beginnig
3.Insert End
4.Insert at given position
5.Display
6.Exit
Enter your choice:5
Elements in linked list are:
0       10      20
1.Create
2.Insert Beginnig
3.Insert End
4.Insert at given position
5.Display
6.Exit
Enter your choice:3
```

```
Enter element:40

1.Create
2.Insert Beginnig
3.Insert End
4.Insert at given position
5.Display
6.Exit
Enter your choice:5
Elements in linked list are:
0       10      20      40
1.Create
2.Insert Beginnig
3.Insert End
4.Insert at given position
5.Display
6.Exit
Enter your choice:4
Enter element:30
Enter position:4

1.Create
2.Insert Beginnig
3.Insert End
4.Insert at given position
5.Display
6.Exit
Enter your choice:5
Elements in linked list are:
0       10      20      30      40
1.Create
2.Insert Beginnig
3.Insert End
4.Insert at given position
5.Display
6.Exit
Enter your choice:6
```

```
1.Create
2.Delete First element
3.Delete last element
4.Delete specific element
5.Display
6.Exit
Enter your choice:1
Enter element:10
If you want to add another element (Y/N):y
Enter element:20
If you want to add another element (Y/N):y
Enter element:30
If you want to add another element (Y/N):y
Enter element:40
If you want to add another element (Y/N):y
Enter element:50
If you want to add another element (Y/N):n

1.Create
2.Delete First element
3.Delete last element
4.Delete specific element
5.Display
6.Exit
Enter your choice:5
Elements in linked list are:
10      20      30      40      50
1.Create
2.Delete First element
3.Delete last element
4.Delete specific element
5.Display
6.Exit
Enter your choice:2

1.Create
2.Delete First element
3.Delete last element
4.Delete specific element
5.Display
6.Exit
Enter your choice:5
Elements in linked list are:
20      30      40      50
```

```
1.Create
2.Delete First element
3.Delete last element
4.Delete specific element
5.Display
6.Exit
Enter your choice:3

1.Create
2.Delete First element
3.Delete last element
4.Delete specific element
5.Display
6.Exit
Enter your choice:5
Elements in linked list are:
20      30      40
1.Create
2.Delete First element
3.Delete last element
4.Delete specific element
5.Display
6.Exit
Enter your choice:4
Enter element to be deleted:30

1.Create
2.Delete First element
3.Delete last element
4.Delete specific element
5.Display
6.Exit
Enter your choice:5
Elements in linked list are:
20      40
1.Create
2.Delete First element
3.Delete last element
4.Delete specific element
5.Display
6.Exit
Enter your choice:6
```

Lab program 7:

1) **WAP to Implement Single Link List with following operations: Sort the linked list, Reverse the linked list, Concatenation of two linked lists.**
2) **WAP to Implement Single Link List to simulate Stack & Queue Operations.**

Code:

1)

```
#include<stdio.h>
#include<stdlib.h>
struct Node{
   int data;
   struct Node *link;
};
typedef struct Node node;
node *start1=NULL,*start2=NULL,*start,*temp,*new1,*pre,*next,*curr,*f,*s=NULL,*t;
void sorting();
void reversing();
void concatenation();
node *create();
void display();
void reverse();
void concatenate();
void sort();
int ch;
char c;
void main(){
   while(1){
      printf("\n1.Sorting a linked list\n2.Reversing a linked list\n3.Concatenation of two linked lists\n4.Exit");
      printf("\nEnter your choice:");
      scanf("%d",&ch);
      switch(ch){
```

```
            case 1:sorting();
                    break;
            case 2:reversing();
                    break;
            case 3:concatenation();
                    break;
            case 4:exit(0);
                    break;
            }
        }
}
void sorting(){
        while(1){
        printf("\n1.Create a linked list\n2.Sorting a linked list\n3.Display\n4.Return\n");
        printf("Enter your choice:");
        scanf("%d",&ch);
        switch(ch){
            case 1:start=create();
                    break;
            case 2:sort();
                    break;
            case 3:printf("After sorting:\n");
                    display();
                    break;
            case 4:return;
                    break;
        }
    }
}
```

```
void reversing(){
   while(1){
      printf("\n1.Create a linked list\n2.Reverse a linked list\n3.Display\n4.Return\n");
      printf("Enter your choice:");
      scanf("%d",&ch);
      switch(ch){
         case 1:start=create();
                break;
         case 2:reverse();
                break;
         case 3:printf("After reversing:\n");
                display();
                break;
         case 4:return;
                break;
      }
   }
}
void concatenation(){
   while(1){
      printf("\n1.Create 1st linked list\n2.Create 2nd linked
list\n3.Concatenate\n4.Display\n5.Return\n");
      printf("Enter your choice:");
      scanf("%d",&ch);
      switch(ch){
         case 1:printf("Enter first linked list");
                start1=create();
                break;
         case 2:printf("Enter second linked list");
                start2=create();
```

```
                break;
            case 3:concatenate();
                break;
            case 4:printf("After concatenation:\n");
                display();
                break;
            case 5:return;
                break;
        }
    }
}
node *create(){
    start=NULL;
    do{
        new1=(node*)malloc(sizeof(node));
        printf("Enter value:");
        scanf("%d",&new1->data);
        if(start==NULL){
            start=new1;
            curr=new1;
        }else{
            curr->link=new1;
            curr=new1;
        }
        printf("Do you want to add another element(Y/N):");
        scanf("%s",&c);
    }while(c == 'y' || c == 'Y');
    curr->link=NULL;
    return(start);
```

```
}
void display(){
   if(start==NULL){
      printf("Linked list is empty");
      return;
   }
   temp=start;
   printf("Elements in the linked list are:\n");
   while(temp!=NULL){
      printf("%d\t",temp->data);
      temp=temp->link;
   }
}
void reverse(){
   f=start;
   while(f!=NULL){
      t=s;
      s=f;
      f=f->link;
      s->link=t;
   }
   start=s;
}
void concatenate(){
   if(start1==NULL && start2==NULL){
      printf("Linked list is empty");
      return;
   }
   if(start1==NULL){
```

```
        start=start2;
        return;
    }
    if(start2==NULL){
        start=start1;
        return;
    }
    temp=start1;
    while(temp->link!=NULL){
        temp=temp->link;
    }
    temp->link=start2;
    start=start1;
    start1=NULL;
    start2=NULL;
}
void sort(){
    int n=0;
    temp=start;
    while(temp!=NULL){
        n=n+1;
        temp=temp->link;
    }
    for(int i=0;i<n;i++){
        pre=start;
        next=start->link;
        for(int j=0;j<n-i-1;j++){
            if(pre->data > next->data){
                int d=pre->data;
```

```
            pre->data=next->data;
            next->data=d;
          }
          pre=next;
          next=next->link;
        }
    }
}
```

2)

```
#include<stdio.h>
#include<stdlib.h>
struct Node{
    int data;
    struct Node *link;
};
typedef struct Node node;
node *front=NULL,*rear=NULL,*temp,*new1,*pre,*next,*top=NULL;
void queues();
void insert();
void delete();
void display1();
void stacks();
void push();
void pop();
void display2();
int ch;
void main(){
    while(1){
        printf("\n1.Linked implementation of queues\n2.Linked implementation of
stacks\n3.Exit\n");
```

```
        printf("Enter your choice:");
        scanf("%d",&ch);
        switch(ch){
           case 1:queues();
                  break;
           case 2:stacks();
                  break;
           case 3:exit(0);
                  break;
           }
        }
}
void queues(){
        while(1){
        printf("\n1.Insert\n2.Delete\n3.Display\n4.Return\n");
        printf("Enter your choice:");
        scanf("%d",&ch);
        switch(ch){
           case 1:insert();
                  break;
           case 2:delete();
                  break;
           case 3:display1();
                  break;
           case 4:return;
                  break;
           }
        }
}
```

```
void stacks(){
        while(1){
        printf("\n1.Push\n2.Pop\n3.Display\n4.Return\n");
        printf("Enter your choice:");
        scanf("%d",&ch);
        switch(ch){
           case 1:push();
                  break;
           case 2:pop();
                  break;
           case 3:display2();
                  break;
           case 4:return;
                  break;
           }
        }
}
void push(){
   new1=(node*)malloc(sizeof(node));
   printf("Enter element:");
   scanf("%d",&new1->data);
   if(top==NULL){
      top=new1;
      new1->link=NULL;
      return;
   }
   new1->link=top;
   top=new1;
}
```

```
void pop(){
    if(top==NULL){
        printf("Linked list is empty");
        return;
    }
    temp=top;
    top=top->link;
    free(temp);
}
void display2(){
    if(top==NULL){
        printf("Linked list is empty");
        return;
    }
    printf("Elements in linked list are:\n");
    temp=top;
    while(temp!=NULL){
        printf("%d\t",temp->data);
        temp=temp->link;
    }
}
void insert(){
    new1=(node*)malloc(sizeof(node));
    printf("Enter element:");
    scanf("%d",&new1->data);
    if(front==NULL && rear==NULL){
        front=new1;
        rear=new1;
        new1->link=NULL;
```

```
        return;
    }
    temp=rear;
    while((temp->link)!=NULL){
        temp=temp->link;
    }
    temp->link=new1;
    new1->link=NULL;
}
void delete(){
    if(front==NULL && rear==NULL){
        printf("Linked list is empty");
        return;
    }
    temp=front;
    front=front->link;
    free(temp);
}

void display1(){
    if(front==NULL){
        printf("Linked list is empty");
        return;
    }
    temp=front;
    printf("Elements in the linked list are:\n");
    while(temp!=NULL){
        printf("%d\t",temp->data);
        temp=temp->link;
```

```
        }
    }
```

Output:

```
1.Sorting a linked list
2.Reversing a linked list
3.Concatenation of two linked lists
4.Exit
Enter your choice:1

1.Create a linked list
2.Sorting a linked list
3.Display
4.Return
Enter your choice:1
Enter value:10
Do you want to add another element(Y/N):y
Enter value:1
Do you want to add another element(Y/N):y
Enter value:5
Do you want to add another element(Y/N):y
Enter value:20
Do you want to add another element(Y/N):y
Enter value:15
Do you want to add another element(Y/N):n

1.Create a linked list
2.Sorting a linked list
3.Display
4.Return
Enter your choice:3
After sorting:
Elements in the linked list are:
10      1       5       20      15
1.Create a linked list
2.Sorting a linked list
3.Display
4.Return
Enter your choice:2

1.Create a linked list
2.Sorting a linked list
3.Display
4.Return
Enter your choice:3
After sorting:
Elements in the linked list are:
1       5       10      15      20
```

```
1.Create a linked list
2.Sorting a linked list
3.Display
4.Return
Enter your choice:4

1.Sorting a linked list
2.Reversing a linked list
3.Concatenation of two linked lists
4.Exit
Enter your choice:2

1.Create a linked list
2.Reverse a linked list
3.Display
4.Return
Enter your choice:1
Enter value:10
Do you want to add another element(Y/N):y
Enter value:20
Do you want to add another element(Y/N):y
Enter value:30
Do you want to add another element(Y/N):y
Enter value:40
Do you want to add another element(Y/N):n

1.Create a linked list
2.Reverse a linked list
3.Display
4.Return
Enter your choice:3
After reversing:
Elements in the linked list are:
10      20      30      40
1.Create a linked list
2.Reverse a linked list
3.Display
4.Return
Enter your choice:2

1.Create a linked list
2.Reverse a linked list
3.Display
4.Return
Enter your choice:3
```

```
After reversing:
Elements in the linked list are:
40      30      20      10
1.Create a linked list
2.Reverse a linked list
3.Display
4.Return
Enter your choice:4

1.Sorting a linked list
2.Reversing a linked list
3.Concatenation of two linked lists
4.Exit
Enter your choice:3

1.Create 1st linked list
2.Create 2nd linked list
3.Concatenate
4.Display
5.Return
Enter your choice:1
Enter first linked listEnter value:10
Do you want to add another element(Y/N):y
Enter value:20
Do you want to add another element(Y/N):y
Enter value:30
Do you want to add another element(Y/N):n

1.Create 1st linked list
2.Create 2nd linked list
3.Concatenate
4.Display
5.Return
Enter your choice:2
Enter second linked listEnter value:40
Do you want to add another element(Y/N):y
Enter value:50
Do you want to add another element(Y/N):n

1.Create 1st linked list
2.Create 2nd linked list
3.Concatenate
4.Display
5.Return
Enter your choice:3
```

```
1.Create 1st linked list
2.Create 2nd linked list
3.Concatenate
4.Display
5.Return
Enter your choice:4
After concatenation:
Elements in the linked list are:
10      20      30      40      50
1.Create 1st linked list
2.Create 2nd linked list
3.Concatenate
4.Display
5.Return
Enter your choice:4
After concatenation:
Elements in the linked list are:
10      20      30      40      50
1.Create 1st linked list
2.Create 2nd linked list
3.Concatenate
4.Display
5.Return
Enter your choice:5

1.Sorting a linked list
2.Reversing a linked list
3.Concatenation of two linked lists
4.Exit
Enter your choice:4
```

```
1.Linked implementation of queues
2.Linked implementation of stacks
3.Exit
Enter your choice:1

1.Insert
2.Delete
3.Display
4.Return
Enter your choice:1
Enter element:10

1.Insert
2.Delete
3.Display
4.Return
Enter your choice:1
Enter element:20

1.Insert
2.Delete
3.Display
4.Return
Enter your choice:3
Elements in the linked list are:
10      20
1.Insert
2.Delete
3.Display
4.Return
Enter your choice:2

1.Insert
2.Delete
3.Display
4.Return
Enter your choice:3
Elements in the linked list are:
20
1.Insert
2.Delete
3.Display
4.Return
Enter your choice:4
```

```
1.Linked implementation of queues
2.Linked implementation of stacks
3.Exit
Enter your choice:2

1.Push
2.Pop
3.Display
4.Return
Enter your choice:1
Enter element:10

1.Push
2.Pop
3.Display
4.Return
Enter your choice:1
Enter element:20

1.Push
2.Pop
3.Display
4.Return
Enter your choice:3
Elements in linked list are:
20      10
1.Push
2.Pop
3.Display
4.Return
Enter your choice:2

1.Push
2.Pop
3.Display
4.Return
Enter your choice:3
Elements in linked list are:
10
1.Push
2.Pop
3.Display
4.Return
Enter your choice:4
```

```
1.Linked implementation of queues
2.Linked implementation of stacks
3.Exit
Enter your choice:3
```

Lab program 8:

Program to Implement Binary Search Tree

Code:

```
#include<stdio.h>
struct Node{
   struct Node *left;
   int data;
   struct Node *right;
};
typedef struct Node node;
node *new1,*curr,*root,*ptr;
void create_bst();
void preorder();
void inorder();
void postorder();
int ch,item;
char c;
void main(){
   while(1){
      printf("\n1.Create a binary search tree\n2.Traverse using Preoder\n3.Traverse using Inorder\n4.Traverse using Postorder\n5.Exit\n");
      printf("Enter your choice:");
      scanf("%d",&ch);
      switch(ch){
         case 1:create_bst();
         break;
         case 2:{printf("\nAfter traversing using Preorder:\n");
               preorder(root);
               break;}
         case 3:{printf("\nAfter traversing using Inorder:\n");
```

```
                inorder(root);
                break;}
        case 4:{printf("\nAfter traversing using postorder:\n");
                postorder(root);
                break;}
        case 5:exit(0);
        break;
      }
   }
}
void create_bst(){
   new1=(node*)malloc(sizeof(node));
   printf("Enter data:");
   scanf("%d",&new1->data);
   new1->left=NULL;
   new1->right=NULL;
   root=new1;
   while(1){
      printf("Do you want to add another element (Y/N):");
      scanf("%s",&c);
      if(c=='y' || c=='Y'){
         new1=(node*)malloc(sizeof(node));
         printf("Enter data:");
         scanf("%d",&new1->data);
         item=new1->data;
         new1->left=NULL;
         new1->right=NULL;
         curr=root;
         while(curr!=NULL){
```

```
            ptr=curr;
            curr=(item>curr->data)?curr->right:curr->left;
        }
        if(item<ptr->data){
            ptr->left=new1;
        }
        else{
            ptr->right=new1;
        }
    }
    else{
        return;
    }
  }
}
void preorder(node *temp){
  if(temp!=NULL){
    printf("%d\t",temp->data);
    preorder(temp->left);
    preorder(temp->right);
  }
}
void inorder(node *temp){
  if(temp!=NULL){
    inorder(temp->left);
    printf("%d\t",temp->data);
    inorder(temp->right);
  }
}
```

```
void postorder(node *temp){
   if(temp!=NULL){
      postorder(temp->left);
      postorder(temp->right);
      printf("%d\t",temp->data);
   }
}
```

Output:

```
1.Create a binary search tree
2.Traverse using Preoder
3.Traverse using Inorder
4.Traverse using Postorder
5.Exit
Enter your choice:1
Enter data:10
Do you want to add another element (Y/N):y
Enter data:35
Do you want to add another element (Y/N):y
Enter data:15
Do you want to add another element (Y/N):y
Enter data:73
Do you want to add another element (Y/N):y
Enter data:28
Do you want to add another element (Y/N):y
Enter data:99
Do you want to add another element (Y/N):y
Enter data:57
Do you want to add another element (Y/N):n

1.Create a binary search tree
2.Traverse using Preoder
3.Traverse using Inorder
4.Traverse using Postorder
5.Exit
Enter your choice:2

After traversing using Preorder:
10      35      15      28      73      57      99
1.Create a binary search tree
2.Traverse using Preoder
3.Traverse using Inorder
4.Traverse using Postorder
5.Exit
Enter your choice:3

After traversing using Inorder:
10      15      28      35      57      73      99
1.Create a binary search tree
2.Traverse using Preoder
3.Traverse using Inorder
4.Traverse using Postorder
5.Exit
Enter your choice:4
```

```
After traversing using postorder:
28      15      57      99      73      35      10
1.Create a binary search tree
2.Traverse using Preoder
3.Traverse using Inorder
4.Traverse using Postorder
5.Exit
Enter your choice:5
```

Lab program 9:

a) **Write a program to traverse a graph using the BFS method.**
b) **Write a program to check whether a given graph is connected or not using the DFS method.**

Code:

a)

```
#include<stdio.h>
int a[10][10],vis[10]={0},q[10],i,j,n,start;
void bfs(int);
void main(){
   printf("Enter number of nodes:");
   scanf("%d",&n);
   printf("Enter adjacency matrix:\n");
   for(i=1;i<=n;i++){
      for(j=1;j<=n;j++){
         scanf("%d",&a[i][j]);
      }
   }
   printf("Enter starting vertex:");
   scanf("%d",&start);
   printf("BFS traversal:\n");
   bfs(start);
}
void bfs(int s){
   int f=0,r=-1;
   vis[s]=1;
   q[++r]=s;
   while(f<=r){
      int curr=q[f++];
      printf("%d\t",curr);
```

```
        for(i=1;i<=n;i++){
            if(a[curr][i]==1 && vis[i]==0){
                vis[i]=1;
                q[++r]=i;
            }
        }
    }
}
```

b)

```
#include<stdio.h>
int a[10][10],vis[10]={0},i,j,k,n;
void dfs(int);
void main(){
    printf("Enter number of nodes:");
    scanf("%d",&n);
    printf("Enter adjacency matrix:\n");
    for(i=1;i<=n;i++){
        for(j=1;j<=n;j++){
            scanf("%d",&a[i][j]);
        }
    }
    dfs(1);
    int con=0;
    for(i=1;i<=n;i++){
        if(vis[i]==1){
            con++;
        }
    }
    if(con==n){
```

```
        printf("Graph is connected");
    }
    else{
        printf("Graph is not connected");
    }
}
void dfs(int v){
    vis[v]=1;
    for(k=1;k<=n;k++){
        if(vis[k]==0 && a[v][k]==1){
            dfs(k);
        }
    }
}
```

Output:

```
Enter number of nodes:4
Enter adjacency matrix:
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
Enter starting vertex:1
BFS traversal:
1       2       3       4
```

```
Enter number of nodes:4
Enter adjacency matrix:
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
Graph is connected
```

```
Enter number of nodes:4
Enter adjacency matrix:
0 1 1 0
1 0 0 0
1 0 0 0
0 0 0 0
Graph is not connected
```

Lab program 10:

Implementation of Linear Probing

Code:

```
#include<stdio.h>
int hashtable[100];
int hashindex,probeindex,m,n,key,data;
void insert_into_hashtable();
void display_hashtable();
int search_in_hashtable();
void main(){
   printf("\nEnter number of memory locations in hash table:");
   scanf("%d",&m);
   for(int i=0;i<m;i++){
      hashtable[i]=-1;
   }
   printf("\nEnter number of keys in hashtable to be placed:");
   scanf("%d",&n);
   insert_into_hashtable();
   display_hashtable();
   printf("\nEnter key to be searched:");
   scanf("%d",&key);
   int res=search_in_hashtable();
   if(res!=-1){
      printf("\nKey %d is found at index %d",key,res);
   }
   else{
      printf("\nKey %d is not found  in hashtable",key);
   }
}
```

```
void insert_into_hashtable(){
    for(int i=0;i<n;i++){
        printf("Enter key to be added to hashtable:");
        scanf("%d",&data);
        hashindex=data % m;
        if(hashtable[hashindex]==-1){
            hashtable[hashindex]=data;
        }
        else{
            while(hashtable[hashindex]!=-1){
                hashindex=(hashindex+1)%m;
            }
            hashtable[hashindex]=data;
        }
    }
}
void display_hashtable(){
    printf("\nHash table:\n");
    for(int i=0;i<m;i++){
        if(hashtable[i]!=-1){
            printf("\nIndex %d --> Key %d",i,hashtable[i]);
        }
        else{
            printf("\nIndex %d --> Empty");
        }
    }
}
int search_in_hashtable(){
    hashindex=key%m;
```

```
    if(hashtable[hashindex]==key){
        return(hashindex);
    }
    else{
        for(int i=0;i<m;i++){
            probeindex=(hashindex+1)%m;
            if(hashtable[probeindex]==key){
                return(probeindex);
            }
            if(hashtable[probeindex]==-1){
                return(-1);
            }
        }
    }
}
```

Output:

```
Enter number of memory locations in hash table:7

Enter number of keys in hashtable to be placed:4
Enter key to be added to hashtable:25
Enter key to be added to hashtable:35
Enter key to be added to hashtable:53
Enter key to be added to hashtable:21

Hash table:

Index 0 --> Key 35
Index 1 --> Key 21
Index 2 --> Empty
Index 3 --> Empty
Index 4 --> Key 25
Index 5 --> Key 53
Index 6 --> Empty
Enter key to be searched:35

Key 35 is found at index 0
```

```
Enter number of memory locations in hash table:7

Enter number of keys in hashtable to be placed:4
Enter key to be added to hashtable:25
Enter key to be added to hashtable:35
Enter key to be added to hashtable:53
Enter key to be added to hashtable:21

Hash table:

Index 0 --> Key 35
Index 1 --> Key 21
Index 2 --> Empty
Index 3 --> Empty
Index 4 --> Key 25
Index 5 --> Key 53
Index 6 --> Empty
Enter key to be searched:22

Key 22 is not found  in hashtable
```