VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

DATA STRUCTURES (23CS3PCDST)

Submitted by

Pramitha J O (1BM23CS240)

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

Sep-2024 to Jan-2025

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled **"DATA STRUCTURES** "carried out by **Pramitha J O (1BM23CS240),** 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. The Lab report has been approved as it satisfies the academic requirements in respect of a Data structures Lab - **(23CS3PCDST)**work prescribed for the said degree.

Prof. Lakshmi Neelima M
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

Index Sheet

Course outcomes:

CO1	Apply the concept of linear and nonlinear data structures.
CO2	Analyse 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.

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

#define MAX 5
int stack[MAX];
int top = -1;

void push(int value) {
   if (top >= MAX - 1) {
      printf("Stack Overflow! Cannot push more elements.\n");
   } else {
      top++;
      stack[top] = value;
      printf("Pushed %d onto the stack.\n", value);
   }
}

void pop() {
```

```
    if (top == -1) {
        printf("Stack Underflow! The stack is empty.\n");
    } else {
        printf("Popped %d from the stack.\n", stack[top]);
        top--;
    }
}

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

int main() {
    int choice, value;

    while (1) {
        printf("\nChoose an operation:\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Display\n");
        printf("4. Exit\n");
```

```
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to push: ");
                scanf("%d", &value);
                push(value);
                break;
            case 2:
                pop();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}
```

```
Choose an operation:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Stack Underflow!

Choose an operation:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Stack is empty.

Choose an operation:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter value to push: 3
Pushed 3 onto the stack.

Choose an operation:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter value to push: 5
Pushed 5 onto the stack.

Choose an operation:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter value to push: 7
```

```
Enter your choice: 2
Popped 1 from the stack.

Choose an operation:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Popped 9 from the stack.

Choose an operation:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Stack elements: 7 5 3

Choose an operation:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 4
```

Lab program 2:

2a) WAP 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).

```
#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>


#define MAX 100


char stack[MAX];

int top = -1;
```

```
int precedence(char operator) {
    if (operator == '+' || operator == '-') {
        return 1;
    } else if (operator == '*' || operator == '/') {
        return 2;
    }
    return 0;
}

int isOperator(char c) {
    return (c == '+' || c == '-' || c == '*' || c == '/');
}

int isOperand(char c) {
    return isalpha(c);
}

void infixToPostfix(char *infix, char *postfix) {
    int i = 0, k = 0;
    char current;

    while (infix[i] != '\0') {
        current = infix[i];

        if (isOperand(current)) {
            postfix[k++] = current;
        }
        else if (current == '(') {
            stack[++top] = current;
```

```
        }
        else if (current == ')') {
            while (top != -1 && stack[top] != '(') {
                postfix[k++] = stack[top--];
            }
            top--;
        }
        else if (isOperator(current)) {
            while (top != -1 && precedence(stack[top]) >= precedence(current)) {
                postfix[k++] = stack[top--];
            }
            stack[++top] = current;
        }

        i++;
    }

    while (top != -1) {
        postfix[k++] = stack[top--];
    }

    postfix[k] = '\0';
}

int main() {
    char infix[MAX], postfix[MAX];

    printf("Enter a valid parenthesized infix expression: ");
    scanf("%s", infix);
```

```
    infixToPostfix(infix, postfix);

    printf("Postfix expression: %s\n", postfix);

    return 0;
}
```

```
Enter a valid parenthesized infix expression: A+B-C/(D^E)*F
Postfix expression: AB+CDE/F*-
```

2b) Move Zeros

```
void moveZeroes(int* nums, int numsSize)
{
    int j=0;
for(int i=0;i<numsSize;i++){
        if(nums[i]!=0){
            int temp=nums[i];
            nums[i]=nums[j];
           nums[j]=temp;
           j++;
        }
    }
}
```

Lab program 3:

3a) WAP to simulate the working of a queue of integers using an array. Provide the following operations: Insert, Delete, Display

The program should print appropriate messages for queue empty and queue overflow conditions.

```
#include<stdio.h>
#include<stdlib.h>
#define SIZE 3
int queue[SIZE], rear=-1,front=-1;
int isEmpty()
{
   if(front==-1)
      return 1;
   return 0;
}
int isFull()
{
   if(rear==SIZE-1)
      return 1;
   return 0;
}
void enqueue(int x)
{
   if(isFull())
      printf("\nqueue overflow");
   else
   {
      if(isEmpty()){
         front=0;
      }
      rear++;
      queue[rear]=x;
      printf("inserted:%d",queue[rear]);
   }
```

```
}
void dequeue()
{
 if(isEmpty())
      printf("\nqueue underflow");
 else
 {
    int m=queue[front];
    if(front==rear)
     front=rear=-1;
    else
     front=front+1;
    printf("\ndeleted element: %d",m);
 }
}
void display()
{
  if(front==-1)
      printf("\n queue is empty");
   else
   {
      for(int i=front;i<=rear;i++)
         printf("  %d  ",queue[i]);
   }
}
void peek()
{

   if(front==-1)
```

```
        printf("\n queue is empty");
    else
        printf("%d",queue[front]);
}
void main()
{
    int x,c;
    while(1)
    {
        printf("\n\n1.Enqueue\n 2.Dequeue\n 3.peek\n 4.Display\n 5.Exit ");
        printf("\nenter choice");
        scanf("%d",&c);
        switch(c)
        {
            case 1: printf(" enter element");
                    scanf("%d",&x);
                    enqueue(x);
                    break;
            case 2: dequeue();
                    break;
            case 3: peek();
                    break;
            case 4: display();
                    break;
            case 5: exit(0);
            default : printf("invalid choice try again");
        }
    }
}
```

```
1.Enqueue
 2.Dequeue
 3.peek
 4.Display
 5.Exit
enter choice1
 enter element3
inserted:3

1.Enqueue
 2.Dequeue
 3.peek
 4.Display
 5.Exit
enter choice1
 enter element6
inserted:6

1.Enqueue
 2.Dequeue
 3.peek
 4.Display
 5.Exit
enter choice4
  3    6

1.Enqueue
 2.Dequeue
 3.peek
 4.Display
 5.Exit
enter choice9
invalid choice try again

1.Enqueue
 2.Dequeue
 3.peek
 4.Display
 5.Exit
enter choice1
 enter element9
```

```
1.Enqueue
 2.Dequeue
 3.peek
 4.Display
 5.Exit
enter choice1
 enter element9

queue overflow

1.Enqueue
 2.Dequeue
 3.peek
 4.Display
 5.Exit
enter choice2

deleted element: 3

1.Enqueue
 2.Dequeue
 3.peek
 4.Display
 5.Exit
enter choice4
  6    9

1.Enqueue
 2.Dequeue
 3.peek
 4.Display
 5.Exit
enter choice2

deleted element: 6

1.Enqueue
 2.Dequeue
 3.peek
 4.Display
 5.Exit
```

```
deleted element: 9

1.Enqueue
 2.Dequeue
 3.peek
 4.Display
 5.Exit
enter choice4

 queue is empty

1.Enqueue
 2.Dequeue
 3.peek
 4.Display
 5.Exit
enter choice5
```

3b) WAP to simulate the working of a circular queue of integers using an array. Provide the following operations: Insert, Delete & Display
The program should print appropriate messages for queue empty and queue overflow condition.

```
#include<stdio.h>

#include<stdlib.h>

#define SIZE 3

int queue[SIZE], rear=-1,front=-1;

int isEmpty()

{

   if(front==-1)

      return 1;

   return 0;

}

int isFull()

{

   if(front==rear+1||front==0&&rear==SIZE-1)

      return 1;
```

```
    return 0;
}
void enqueue(int x)
{
    if(isFull())
        printf("\nqueue overflow");
    else
    {
        if(isEmpty()){
            front=0;
        }
        rear=(rear+1)%SIZE;
        queue[rear]=x;
        printf("inserted:%d",queue[rear]);
    }
}
void dequeue()
{
  if(isEmpty())
        printf("\nqueue underflow");
  else
  {
      int m=queue[front];
      if(front==rear)
        front=rear=-1;
      else
        front=(front+1)%SIZE;
      printf("\ndeleted element: %d",m);
  }
```

```
}
void display()
{
  if(front==-1)
      printf("\n queue is empty");
   else
   { int i;
      for( i=front;i!=rear;i=(i+1)%SIZE)
         printf("  %d  ",queue[i]);
      printf("  %d  ",queue[i]);
}
}
void main()
{
   int x,c;
   while(1)
   {
      printf("\n\n1.Enqueue\n 2.Dequeue\n  3.Display\n 4.Exit ");
      printf("\nenter choice");
      scanf("%d",&c);
      switch(c)
      {
         case 1: printf(" enter element");
               scanf("%d",&x);
               enqueue(x);
               break;
         case 2: dequeue();
               break;
         case 3: display();
```

```
            break;
        case 4: exit(0);
        default : printf("invalid choice try again");
      }
   }
}
```

```
1.Enqueue
 2.Dequeue
  3.Display
 4.Exit
enter choice1
 enter element3
inserted:3

1.Enqueue
 2.Dequeue
  3.Display
 4.Exit
enter choice1
 enter element3
inserted:3

1.Enqueue
 2.Dequeue
  3.Display
 4.Exit
enter choice1
 enter element3
inserted:3

1.Enqueue
 2.Dequeue
  3.Display
 4.Exit
enter choice1
 enter element6

queue overflow

1.Enqueue
 2.Dequeue
  3.Display
 4.Exit
enter choice2
```

```
deleted element: 3

1.Enqueue
 2.Dequeue
  3.Display
 4.Exit
enter choice3
  3    3

1.Enqueue
 2.Dequeue
  3.Display
 4.Exit
enter choice2

deleted element: 3

1.Enqueue
 2.Dequeue
  3.Display
 4.Exit
enter choice1
 enter element5
inserted:5

1.Enqueue
 2.Dequeue
  3.Display
 4.Exit
enter choice1
 enter element7
inserted:7

1.Enqueue
 2.Dequeue
  3.Display
 4.Exit
enter choice3
  3    5    7
```

```
 1.Enqueue
  2.Dequeue
   3.Display
  4.Exit
 enter choice4

 Process returned 0 (0x0)   execution time : 145.544 s
 Press any key to continue.
```

Lab program 4:

4a)WAP to Implement Singly Linked List with following operations

a) Create LinkedList.
b) Insertion of a node at first position, at any position and at end of list.
c) Deletion of first element, specified element and last element in the list
Display the contents of the linked list.

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
struct node
{
   int data;
   struct node *next;
};
struct node *start= NULL;
struct node *create(struct node *start);
struct node *display(struct node *start);
struct node *insert_beg(struct node *start);
struct node *insert_end(struct node *start);
struct node *insert_pos(struct node *start);
struct node *delete_beg(struct node *start);
struct node *delete_end(struct node *start);
struct node *delete_node(struct node *start);
int main()
{
   int ch;
   while(1)
   {
      printf("\n\n1.create \t 2.Display \t 3.Insert beg \t4.Insert end\t5.Insert at pos \t6.
delete beg\t7.delete end\t 8.delete node \t 9.exit");
```

```
        printf("\nenter your choice:");
        scanf("%d",&ch);
        struct node *newnode;
        switch(ch)
        {
           case 1: start=create(start);
                   break;
           case 2: start=display(start);
           break;
           case 3 : start=insert_beg(start);
           break;
           case 4: start=insert_end(start);
           break;
           case 5: start=insert_pos(start);
           break;
           case 6: start=delete_beg(start);
           break;
           case 7: start=delete_end(start);
           break;
           case 8: start=delete_node(start);
           break;
           case 9: exit(0);
            default:printf("invalid");
        }

    }
}
struct node *create(struct node *start)
{
```

```
    struct node *newnode, *ptr;
    int num;
    printf("\nenter -1 to end");
    printf("\nenter num:");
    scanf("%d",&num);
    while(num!=-1)
    {
       newnode=(struct node*)malloc(sizeof(struct node*));
       newnode->data=num;
       newnode->next=NULL;
       if(start==NULL)
          start=newnode;
       else
       {
          ptr=start;
          while(ptr->next!=NULL)
             ptr=ptr->next;
          ptr->next=newnode;
       }
       printf("enter num;");
       scanf("%d",&num);

    }
    return start;

};

struct node *display(struct node *start)
{
```

```
    struct node *ptr;
    ptr=start;
    while(ptr->next!=NULL)
    {
       printf("\n%d",ptr->data);
        ptr=ptr->next;
    }
     printf("\n%d",ptr->data);
    return start;

};

struct node *insert_beg(struct node *start)
{
    struct node *newnode, *ptr;
    int num;
    printf("\nenter num");
   scanf("%d",&num);
   newnode=(struct node*)malloc(sizeof(struct node*));
   newnode->data=num;
   newnode->next=start;
   start=newnode;
   return start;
};
struct node *insert_end(struct node *start)
{
   struct node *newnode, *ptr;
    int num;
    printf("\nenter num:");
```

```
    scanf("%d",&num);
    newnode=(struct node*)malloc(sizeof(struct node*));
    newnode->data=num;
    newnode->next=NULL;
    ptr=start;
    while(ptr->next!=NULL)
       ptr=ptr->next;
    ptr->next=newnode;
    return start;
};
struct node *insert_pos(struct node *start)
{
   struct node *newnode, *ptr,*preptr;
     int num,pos,i=1;

     printf("\nenter num:");
    scanf("%d",&num);
    newnode=(struct node*)malloc(sizeof(struct node*));
    newnode->data=num;
    printf("\nenter the position:");
    scanf("%d",&pos);
     ptr=start;
    while(i!=pos)
    {
          preptr=ptr;
          ptr=ptr->next;
          i++;
    }
    preptr->next=newnode;
```

```
    newnode->next=ptr;
    return start;
};
struct node *delete_beg(struct node *start)
{
    struct node *ptr;
    ptr=start;
    start=start->next;
    free(ptr);
    return start;
};
struct node *delete_end(struct node *start)
{
  struct node *ptr, *preptr;
    ptr=start;
    while(ptr->next!=NULL)
    {
        preptr=ptr;
        ptr=ptr->next;
    }
    preptr->next=NULL;
    free(ptr);
    return start;
};
struct node *delete_node(struct node *start)
{
    struct node *ptr, *preptr;
    int num;
    printf("\nenter num:");
```

```
    scanf("%d",&num);
    ptr=start;
     while(ptr->data!=num)
    {
       preptr=ptr;
       ptr=ptr->next;
    }
    preptr->next=ptr->next;
    free(ptr);
    return start;
};
```

```
1.create        2.Display       3.Insert beg   4.Insert end    5.Insert at pos         6. delete beg   7.delete end    8.delete node    9.exit
enter your choice:1

enter -1 to end
enter num:3
enter num;6
enter num;-1


1.create        2.Display       3.Insert beg   4.Insert end    5.Insert at pos     6. delete beg   7.delete end    8.delete node   9.exit
enter your choice:2

3
6

1.create        2.Display       3.Insert beg   4.Insert end    5.Insert at pos     6. delete beg   7.delete end    8.delete node   9.exit
enter your choice:3

enter num2


1.create        2.Display       3.Insert beg   4.Insert end    5.Insert at pos     6. delete beg   7.delete end    8.delete node   9.exit
enter your choice:4

enter num:7


1.create        2.Display       3.Insert beg   4.Insert end    5.Insert at pos     6. delete beg   7.delete end    8.delete node   9.exit
enter your choice:2

2
3
6
7

1.create        2.Display       3.Insert beg   4.Insert end    5.Insert at pos     6. delete beg   7.delete end    8.delete node   9.exit
enter your choice:5

enter num:4

enter the position:3
```

```
1.create         2.Display       3.Insert beg   4.Insert end    5.Insert at pos     6. delete beg   7.delete end     8.delete node   9.exit
enter your choice:2

2
3
4
6
7

1.create         2.Display       3.Insert beg   4.Insert end    5.Insert at pos     6. delete beg   7.delete end     8.delete node   9.exit
enter your choice:6


1.create         2.Display       3.Insert beg   4.Insert end    5.Insert at pos     6. delete beg   7.delete end     8.delete node   9.exit
enter your choice:7


1.create         2.Display       3.Insert beg   4.Insert end    5.Insert at pos     6. delete beg   7.delete end     8.delete node   9.exit
enter your choice:2

3
4
6

1.create         2.Display       3.Insert beg   4.Insert end    5.Insert at pos     6. delete beg   7.delete end     8.delete node   9.exit
enter your choice:8

enter num:4


1.create         2.Display       3.Insert beg   4.Insert end    5.Insert at pos     6. delete beg   7.delete end     8.delete node   9.exit
enter your choice:2

3
6

1.create         2.Display       3.Insert beg   4.Insert end    5.Insert at pos     6. delete beg   7.delete end     8.delete node   9.exit
enter your choice:9
```

4b) Program - Leetcode platform

```
int majorityElement(int* nums, int n) {
    int vote=0,candidate=-1,i,count=0;
    for(i=0;i<n;i++)
    {
        if(vote==0)
        {
            candidate=nums[i];
            vote=1;
        }
        else
        {
            if(nums[i]==candidate)
                vote++;
            else
                vote--;
        }
```

```
    }
    for(i=0;i<n;i++){
        if(nums[i]==candidate)
            count++;
    }
    if(count>(n/2))
        return candidate;
    return -1;
}
```

Lab program 5:

5a) Program - Leetcode platform

```
* Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
bool isPalindrome(struct ListNode* head)
{  int count=0;
    struct ListNode *ptr;
     ptr=head;
    while(ptr->next!=NULL)
    {
        count++;
        ptr=ptr->next;
```

```
    }
    count++;
    int stack[count],top=-1,flag=0;
    ptr=head;
    while(ptr->next!=NULL)
    {
       stack[++top]=ptr->val;
       ptr=ptr->next;
    }
    stack[++top]=ptr->val;
    ptr=head;
    int temp;
    while(top!=-1)
    {
       temp=stack[top--];
       if(temp!=ptr->val)
       {
          flag=1;
          return false;
       }
       ptr=ptr->next;
    }
    if(flag==0)
       return true;

return 1;
}
```

```
1.create         2.Display       3.Insert_before        4.delete_node   5.exit
enter your choice1

enter -1 to end
enter num:3

enter num:5

enter num:9

enter num:7

enter num:1

enter num:-1


1.create         2.Display       3.Insert_before        4.delete_node   5.exit
enter your choice2
3 5 9 7 1

1.create         2.Display       3.Insert_before        4.delete_node   5.exit
enter your choice4

 enter the val:1


1.create         2.Display       3.Insert_before        4.delete_node   5.exit
enter your choice2
3 5 9 7

1.create         2.Display       3.Insert_before        4.delete_node   5.exit
enter your choice4

 enter the val:9


1.create         2.Display       3.Insert_before        4.delete_node   5.exit
enter your choice2
3 5 7
```

```
1.create         2.Display       3.Insert_before        4.delete_node   5.exit
enter your choice4

 enter the val:3


1.create         2.Display       3.Insert_before        4.delete_node   5.exit
enter your choice2
5 7

1.create         2.Display       3.Insert_before        4.delete_node   5.exit
enter your choice3

enter num:8

 enter the val before you want to insert:5


1.create         2.Display       3.Insert_before        4.delete_node   5.exit
enter your choice2
8 5 7

1.create         2.Display       3.Insert_before        4.delete_node   5.exit
enter your choice5
```

6a) WAP to Implement Single Link List with following operations: Sort the linked list, Reverse the linked list, Concatenation of two linked lists.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   int data;
   struct node *next;
};
struct node *start1=NULL;
struct node *start2=NULL;
struct node *create(struct node *start);
struct node *display(struct node * start);
struct node *sort(struct node *start);
struct node *reverse(struct node *start);
struct node *concat(struct node *start1,struct node *start2);

void main()
{
   int ch,n;
   while(1)
   {
      printf("\n\n1.create\t2.display\t3.sort\t4.reverse\t5.concat\t6.exit\n");
      printf("\nenter choice");
      scanf("%d",&ch);
      switch(ch)
      {
         case 1: printf("\nlist 1");
                 start1=create(start1);
                 printf("\nlist 2");
```

```
                start2=create(start2);
                break;
        case 2: printf("\nlist 1");
                start1=display(start1);
                printf("\nlist 2");
               start2=display(start2);
               break;
        case 3: start1=sort(start1);
                start2=sort(start2);
                break;
        case 4: start1=reverse(start1);
                start2=reverse(start2);
                break;
        case 5 : start1=concat(start1,start2);
                   break;
        case 6: exit(0);
        default : printf("invalid");
      }
   }
}

struct node *create(struct node *start)
{
   struct node *newnode, *ptr;
   int num;
   printf("\nenter -1 to end");
   printf("\nenter num");
   scanf("%d",&num);
   while(num!=-1)
```

```
    {
       newnode=(struct node*)malloc(sizeof(struct node*));
       newnode->data=num;
       newnode->next=NULL;
       if(start==NULL)
          start=newnode;
       else
       {
          ptr=start;
          while(ptr->next!=NULL)
             ptr=ptr->next;
          ptr->next=newnode;
       }
       printf("\nenter num");
       scanf("%d",&num);

    }
    return start;

};

struct node *display(struct node *start)
{
     struct node *ptr;
     ptr=start;
     while(ptr->next!=NULL)
     {
         printf("\n%d",ptr->data);
          ptr=ptr->next;
```

```
    }
     printf("\n%d",ptr->data);
    return start;

};

struct node *sort(struct node *start)
{
   struct node *ptr1,*ptr2;
   ptr1=start;
   int temp;
   while(ptr1->next!=NULL)
   {
      ptr2=ptr1->next;
      while(ptr2!=NULL)
      {
         if(ptr1->data>ptr2->data)
         {
            temp=ptr1->data;
            ptr1->data=ptr2->data;
            ptr2->data=temp;

         }
         ptr2=ptr2->next;
      }
      ptr1=ptr1->next;
   }
   return start;
```

```
}

struct node *reverse(struct node *start)
{
   struct node *prev=NULL;
   struct node *ptr=start;
   struct node *next=NULL;
   while(ptr!=NULL)
   {
      next=ptr->next;
      ptr->next=prev;
      prev=ptr;
      ptr=next;
   }
   start=prev;
   return start;
}
struct node *concat(struct node *start1, struct node *start2)
 {
    struct node *ptr;
    ptr=start1;
    if(start1==NULL)
    {
      start1=start2;
    }
    else
    {
       while(ptr->next!=NULL)
       {
```

```
        ptr=ptr->next;
      }
      ptr->next=start2;
    }
    return start1;
}
```

```
1.create        2.display       3.sort  4.reverse       5.concat        6.exit

enter choice1

list 1
enter -1 to end
enter num3

enter num8

enter num1

enter num-1

list 2
enter -1 to end
enter num7

enter num9

enter num2

enter num-1


1.create        2.display       3.sort  4.reverse       5.concat        6.exit

enter choice2

list 1
3
8
1
list 2
7
9
2

1.create        2.display       3.sort  4.reverse       5.concat        6.exit
```

```
1.create        2.display       3.sort  4.reverse       5.concat        6.exit

enter choice3


1.create        2.display       3.sort  4.reverse       5.concat        6.exit

enter choice2

list 1
1
3
8
list 2
2
7
9

1.create        2.display       3.sort  4.reverse       5.concat        6.exit

enter choice4


1.create        2.display       3.sort  4.reverse       5.concat        6.exit

enter choice2

list 1
8
3
1
list 2
9
7
2

1.create        2.display       3.sort  4.reverse       5.concat        6.exit

enter choice5
```

```
enter choice5


1.create        2.display       3.sort  4.reverse       5.concat        6.exit

enter choice2

list 1
8
3
1
9
7
2
list 2
9
7
2

1.create        2.display       3.sort  4.reverse       5.concat        6.exit

enter choice6
```

6b) WAP to Implement Single Link List to simulate Stack & Queue Operations.

```
#include<stdlib.h>
struct node
{
   int data;
   struct node *next;
};
struct node *top=NULL;
void push(int num)
{
 struct node *newnode;
 newnode=(struct node*)malloc(sizeof(struct node*));
 newnode->data=num;
 newnode->next=NULL;

 if(top==NULL)
 {
    newnode->next=NULL;
    top=newnode;

 }
 else
 {
    newnode->next=top;
    top=newnode;


 }
}
```

```
void pop()
{
   if(top==NULL)
   {
      printf("\nStack Underflow");
   }
   else
   {
      printf("deleted:%d", top->data);
      top=top->next;
   }
}

void display()
{
   if(top==NULL)
   {
      printf("\nStack Empty");
   }
   else
   {
      struct node *ptr;
      ptr=top;
      while(ptr!=NULL)
      {
         printf("%d\t",ptr->data);
         ptr=ptr->next;
      }
```

```
    }
}



void main()
{
    int n,c;
    while(1)
    {
      printf("\n\n1.push\t 2.pop\t 3.display\t 4.exit\n your choice:");
      scanf("%d",&c);
      switch(c)
      {
         case 1: printf("\nenter element:");
                 scanf("%d",&n);
                 push(n);
                 break;
          case 2:pop();
          break;
          case 3: display();
          break;
          case 4: exit(0);
          default: printf("\ninvalid choice");
      }
    }
}
```

```
1.push   2.pop   3.display       4.exit
 your choice:2

Stack Underflow

1.push   2.pop   3.display       4.exit
 your choice:3

Stack Empty

1.push   2.pop   3.display       4.exit
 your choice:1

enter element:3


1.push   2.pop   3.display       4.exit
 your choice:1

enter element:5


1.push   2.pop   3.display       4.exit
 your choice:1

enter element:7


1.push   2.pop   3.display       4.exit
 your choice:3
7       5       3

1.push   2.pop   3.display       4.exit
 your choice:2
deleted:7

1.push   2.pop   3.display       4.exit
 your choice:2
deleted:5
```

```
 your choice:1

enter element:7


1.push   2.pop   3.display       4.exit
 your choice:3
7       5       3

1.push   2.pop   3.display       4.exit
 your choice:2
deleted:7

1.push   2.pop   3.display       4.exit
 your choice:2
deleted:5

1.push   2.pop   3.display       4.exit
 your choice:3
3

1.push   2.pop   3.display       4.exit
 your choice:2
deleted:3

1.push   2.pop   3.display       4.exit
 your choice:2

Stack Underflow

1.push   2.pop   3.display       4.exit
 your choice:3

Stack Empty

1.push   2.pop   3.display       4.exit
 your choice:4

Process returned 0 (0x0)   execution time : 95.619 s
Press any key to continue.
```

```
#include<stdio.h>

#include<stdlib.h>

struct node

{

   int data;
```

```
    struct node *next;
};
struct node *front=NULL,*rear=NULL;

void insert(int num)
{
    struct node *newnode;
    newnode=(struct node*)malloc(sizeof(struct node*));
    newnode->data=num;
    newnode->next=NULL;
    if(front==NULL)
    {
        front=newnode;
        rear=newnode;
    }
    else
    {
        rear->next=newnode;
        rear=newnode;
    }
}

void Delete()
{
    if(front==NULL)
    {
        printf("queue Underflow");
    }
    else
```

```
    {
        printf("\ndeleted:%d",front->data);
        front=front->next;
    }
}

void display()
{
    if(front==NULL)
    {
        printf("\nqueue Empty");
    }
    else
    {
        struct node *ptr;
        ptr=front;
        while(ptr!=NULL)
        {
            printf("%d  ",ptr->data);
            ptr=ptr->next;
        }
    }

}

void main()
{
    int n,c;
    while(1)
```

```
  {
    printf("\n\n1.insert\t 2.delete\t 3.display\t 4.exit\n your choice:");
    scanf("%d",&c);
    switch(c)
    {
       case 1: printf("\nenter element:");
            scanf("%d",&n);
            insert(n);
            break;
        case 2:Delete();
        break;
        case 3: display();
        break;
        case 4: exit(0);
        default: printf("\ninvalid choice");
    }
  }
}
```

```
1.insert         2.delete         3.display       4.exit
 your choice:2
queue Underflow

1.insert         2.delete         3.display       4.exit
 your choice:3

queue Empty

1.insert         2.delete         3.display       4.exit
 your choice:1

enter element:3


1.insert         2.delete         3.display       4.exit
 your choice:1

enter element:5


1.insert         2.delete         3.display       4.exit
 your choice:1

enter element:7


1.insert         2.delete         3.display       4.exit
 your choice:3
3  5  7

1.insert         2.delete         3.display       4.exit
 your choice:2

deleted:3

1.insert         2.delete         3.display       4.exit
 your choice:2

deleted:5
```

```
deleted:5

1.insert         2.delete         3.display       4.exit
 your choice:3
7

1.insert         2.delete         3.display       4.exit
 your choice:2

deleted:7

1.insert         2.delete         3.display       4.exit
 your choice:2
queue Underflow

1.insert         2.delete         3.display       4.exit
 your choice:3

queue Empty

1.insert         2.delete         3.display       4.exit
 your choice:4
```

Lab program 7:

7a) WAP to Implement doubly link list with primitive operations

a) Create a doubly linked list.

b) Insert a new node to the left of the node.

c) Delete the node based on a specific value

d) Display the contents of the list

7b) Program - Leetcode platform

* Complete the 'twoStacks' function below.

```
 * The function is expected to return an INTEGER.
 * The function accepts following parameters:
 *  1. INTEGER maxSum
 *  2. INTEGER_ARRAY a
 *  3. INTEGER_ARRAY b


int twoStacks(int maxSum, int n, int* a, int m, int* b) {

    int current_sum_a = 0;
    int count_a = 0;

    for (int i = 0; i < n; i++) {
        if (current_sum_a + a[i] <= maxSum) {
            current_sum_a += a[i];
            count_a++;
        } else {
            break;
        }
    }


    int max_count = count_a;
    int current_sum_b = 0;


    for (int j = 0; j < m; j++) {
        current_sum_b += b[j];
        while (current_sum_a + current_sum_b > maxSum && count_a > 0) {
```

```
            count_a--;
            current_sum_a -= a[count_a];
        }

        if (current_sum_a + current_sum_b <= maxSum) {
            max_count = (max_count > count_a + (j + 1)) ? max_count : (count_a + (j + 1));
        }
    }

    return max_count;

}
```

Lab program 8:

8a) Write a program

a) To construct a binary Search tree.

b) To traverse the tree using all the methods i.e., in-order, preorder and post order

c) To display the elements in the tree.

```
#include<stdio.h>
#include<stdlib.h>
typedef struct bst
{
   int data;
   struct bst *left, *right;
}node;

node *create()
{
```

```
    node *temp;
    temp=(node *)malloc(sizeof(node*));
    printf("enter data");
    scanf("%d", &temp->data);
    temp->left=NULL;
    temp->right=NULL;
    return temp;
}

node *insert(node *root, node *temp)
{
    if(temp->data<root->data)
    {
      if(root->left!=NULL)
      insert(root->left, temp);
      else
        root->left=temp;

    }
    if(temp->data>root->data)
    {
        if(root->right!=NULL)
            insert(root->right, temp);
        else
            root->right=temp;
    }
}
```

```
void inorder( node *root)
{
   if(root!=NULL)
   {
      inorder(root->left);
      printf("%d  ", root->data);
      inorder(root->right);
   }

}

void preorder( node *root)
{
   if(root!=NULL)
   {
      printf("%d  ", root->data);
      preorder(root->left);
      preorder(root->right);
   }

}

void postorder( node *root)
{
   if(root!=NULL)
   {
      postorder(root->left);
      postorder(root->right);
```

```
        printf("%d  ", root->data);
    }

}


void main()
{

    char ch;
    node *root=NULL, *temp;
    do
    {
        temp=create();
        if(root==NULL)
            root=temp;
        else
            insert(root, temp);
        printf("\nyou want to enter more number(y/n)");
        getchar();
        scanf("%c",&ch);
    }while(ch=='y'||ch=='Y');
    printf("\n Preorder:");
    preorder(root);
     printf("\n Inorder:");
    inorder(root);
     printf("\n Postorder:");
    postorder(root);
```

}

```
enter data20

you want to enter more number(y/n)y
enter data25

you want to enter more number(y/n)y
enter data15

you want to enter more number(y/n)y
enter data10

you want to enter more number(y/n)y
enter data5

you want to enter more number(y/n)y
enter data35

you want to enter more number(y/n)n

 Preorder:20  15  10  5  25  35
 Inorder:5  10  15  20  25  35
 Postorder:5  10  15  35  25  20
Process returned 4 (0x4)   execution time : 71.718 s
Press any key to continue.
```

8b) Program - Leetcode platform

```
bool hasPathSum(struct TreeNode* root, int targetSum) {
    if (root == NULL) {
        return false;
    }

    if (root->left == NULL && root->right == NULL) {
        return targetSum == root->val;
    }

    int remainingSum = targetSum - root->val;
    return hasPathSum(root->left, remainingSum) || hasPathSum(root->right, remainingSum);
}
```

Lab program 9:

9a) Write a program to traverse a graph using BFS method.

```
#include<stdio.h>
#include<stdlib.h>
int a[20][20],q[20], visited[20],n,i,j,f=0,r=-1;
void bfs(int v)
{
   for(i=1;i<=n;i++)
   {
      if(a[v][i]&& !visited[i])
         q[++r]=i;
   }
   if(f<=r)
   {
      visited[q[f]]=1;
      bfs(q[f++]);
   }
}

void main()
{
   int v;
   printf("\nenter no of vertices:");
   scanf("%d",&n);
   for(i=1;i<=n;i++)
   {
      q[i]=0;
      visited[i]=0;
```

```
    }
    printf("\nenter graph data");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
            scanf("%d", &a[i][j]);
    }
    printf("\nenter starting vertex");
    scanf("%d",&v);
    bfs(v);
    printf("\nnodes which are reachable:");
    for(i=1;i<=n;i++)
    {
        if(visited[i])
            printf("%d\t", i);

    }
}
```

```
enter no of vertices:6

enter graph data0 1 1 1 1 0
1 0 0 1 0 0
1 0 0 1 1 0
1 1 1 0 0 0
1 0 1 0 0 0
0 0 0 0 0 0

enter starting vertex4

nodes which are reachable:1     2       3       4       5
Process returned 6 (0x6)   execution time : 196.463 s
Press any key to continue.
```

9b) Write a program to check whether given graph is connected or not using DFS method.

```
#include<stdio.h>
#include<stdlib.h>
int a[20][20],s[20],n;
void dfs(int v)//v=vertex
{
   s[v]=1;
   int i;
   for(i=1;i<=n;i++)
   {
   if(a[v][i]&&!s[i])
   {
      printf("\n%d->%d",v,i);
      dfs(i);
   }
   }
}

void main()
{
   int i,j,count=0;
   printf("\n enter the no. of vertices:");
   scanf("%d",&n);
   for(i=1;i<=n;i++)
   {
      s[i]=0;
      for(j=1;j<=n;j++)
         a[i][j]=0;
   }
```

```
    printf("\n Enter the Adjacent matrix:\n");
    for(i=1;i<=n;i++)
    {
       for(j=1;j<=n;j++)
          scanf("%d",&a[i][j]);
    }
    dfs(1);
    printf("\n");

    for(int i=1;i<=n;i++)
    {
       if(s[i])
          count++;
    }
    if(count==n)
       printf("Graph is Connected");
    else
       printf("Graph is disconnected");

}
```

```
 enter the no. of vertices:6

 Enter the Adjacent matrix:
0 1 1 1 1 0
1 0 0 1 0 0
1 0 0 1 1 1
1 1 1 0 0 1
1 0 1 0 0 0
0 0 1 1 0 0

1->2
2->4
4->3
3->5
3->6
Graph is Connected
Process returned 18 (0x12)   execution time : 80.868 s
Press any key to continue.
```

Lab program 10:

Given a File of N employee records with a set K of Keys(4- digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table (HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT.

Let the keys in K and addresses in L are integers.

Design and develop a Program in C that uses Hash function H: K -> L as H(K)=K mod m (remainder method), and implement hashing technique to map a given key K to the address space L.

Resolve the collision (if any) using linear probing.

```
#include <stdio.h>

#include <stdlib.h>


#define TABLE_SIZE 100 // Size of the hash table (m memory locations)


// Structure to represent an employee record

typedef struct {

   int key;    // 4-digit key

   char name[30];
```

```
    int age;
} Employee;

// Hash table array
Employee* hashTable[TABLE_SIZE];

// Initialize the hash table
void initializeHashTable() {
    for (int i = 0; i < TABLE_SIZE; i++) {
        hashTable[i] = NULL;
    }
}

// Hash function to compute memory address
int hashFunction(int key) {
    return key % TABLE_SIZE;
}

// Insert an employee record into the hash table
void insert(Employee* emp) {
    int index = hashFunction(emp->key);

    // Linear probing to handle collisions
    while (hashTable[index] != NULL) {
        index = (index + 1) % TABLE_SIZE; // Go to the next index
    }

    hashTable[index] = emp;
}
```

```
// Search for an employee record by key
Employee* search(int key) {
    int index = hashFunction(key);

    // Search using linear probing
    while (hashTable[index] != NULL) {
        if (hashTable[index]->key == key) {
            return hashTable[index];
        }
        index = (index + 1) % TABLE_SIZE;
    }

    return NULL; // Record not found
}

// Display the hash table
void displayHashTable() {
    printf("\nHash Table:\n");
    for (int i = 0; i < TABLE_SIZE; i++) {
        if (hashTable[i] != NULL) {
            printf("Index %d: Key: %d, Name: %s, Age: %d\n", i, hashTable[i]->key, hashTable[i]-
>name, hashTable[i]->age);
        }
    }
}

// Main function
int main() {
    initializeHashTable();
```

```
    int n;
    printf("Enter the number of employees: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        Employee* emp = (Employee*)malloc(sizeof(Employee));

        printf("Enter key (4-digit), name, and age for employee %d: ", i + 1);
        scanf("%d %s %d", &emp->key, emp->name, &emp->age);

        insert(emp);
    }

    displayHashTable();

    int searchKey;
    printf("\nEnter a key to search for: ");
    scanf("%d", &searchKey);

    Employee* result = search(searchKey);
    if (result != NULL) {
        printf("Employee found: Key: %d, Name: %s, Age: %d\n", result->key, result->name,
result->age);
    } else {
        printf("Employee with key %d not found.\n", searchKey);
    }

    return 0;
```

```
}
Enter the number of employees: 3
Enter key (4-digit), name, and age for employee 1: 1234 prami 20
Enter key (4-digit), name, and age for employee 2: 5678 rakshi 21
Enter key (4-digit), name, and age for employee 3: 3579 sahana 19

Hash Table:
Index 34: Key: 1234, Name: prami, Age: 20
Index 78: Key: 5678, Name: rakshi, Age: 21
Index 79: Key: 3579, Name: sahana, Age: 19

Enter a key to search for: 3579
Employee found: Key: 3579, Name: sahana, Age: 19

Process returned 0 (0x0)   execution time : 315.741 s
Press any key to continue.
```