**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**

****

**LAB REPORT**

**On**

**DATA STRUCTURES (23CS3PCDST)**

**Submitted by**

**ARPITH GOWDA H S (1BM23CS053)**

**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 ARPITH GOWDA H S**(1BM23CS053)**, 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.Sneha S Bagalkot** **Dr. Kavitha Sooda**

Assistant Professor Professor and Head

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

**Index Sheet**

|  |  |  |
| --- | --- | --- |
| **Sl.**  **No.** | **Experiment Title** | **Page No.** |
| 1 | IMPLEMENTATION OF STACK OPERATIONS | 4 TO 6 |
| 2 | IMPLEMENTATION OF INFIX TO POSTFIX | 7 TO 9 |
| 3 | IMPLEMENTING QUEUE OPERATIONS AND LEETCODE PROBLEM | 9 TO 11 |
| 4 | IMPLEMENTING CIRCULAR QUEUE OPERATIONS AND LEETCODE CHALLENGE | 12 TO 15 |
| 5 | FACTORIAL, FIBONACCI, TOWER OF HANOI USING RECURSION AND LEETCODE CHALLENGE | 15 TO 19 |
| 6 | CREATING, INSERTING, DELETING AND DISPLAYING LINKED LIST | 19 TO 30 |
| 7 | CONCATENATION, REVERSING AND SORTING OF LINKED LIST | 30 TO 36 |
| 8 | IMPLEMENT SINGLE LINKED LIST TO SIMULATE STACK AND QUEUE OPERATIONS | 37 TO 46 |
| 9 | DOUBLY LINKED LIST AND IMPLEMENTING BINARY SEARCH | 46 TO 61 |
| 10 | DFS AND BFS TRAVERSAL | 62 TO 65 |

**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:**

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<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:3**

**Stack contents:**

**30**

**20**

**10**

**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:2**

**Popped element is 20**

**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:2**

**STACK UNDERFLOW**

**1:Push**

**2:Pop**

**3:Display**

**4:Exit**

**Enter your choice:4 \*/**

**2)Program for DS lab on 08/10/2024Write 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)**

**#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;**

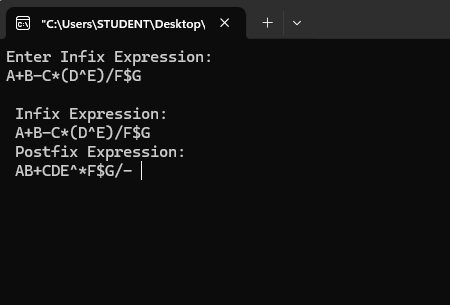
**case '#':p=20;**

**break;**

**}**

**return(p);**

**}**



**3)Implementation of Queue Operations and Leetcode problem on Clearing Digits**

**#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++)**

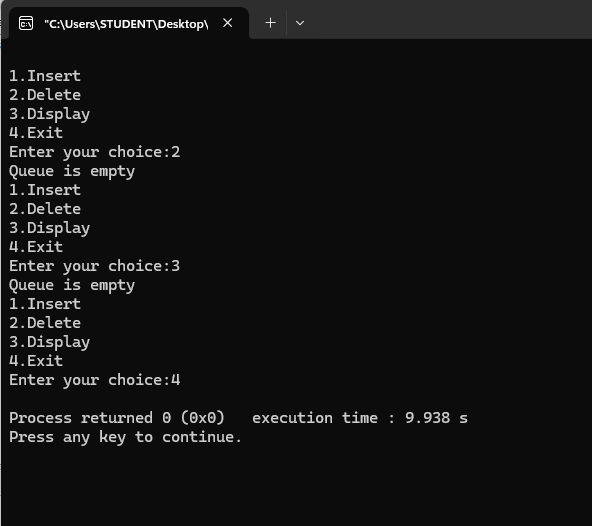
**{**

**printf("%d\t",q[i]);**

**}**

**return;**

**}**



**4)Programs for Circular Queue implementation and Leetcode challenge.**

**#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++)**

**{**

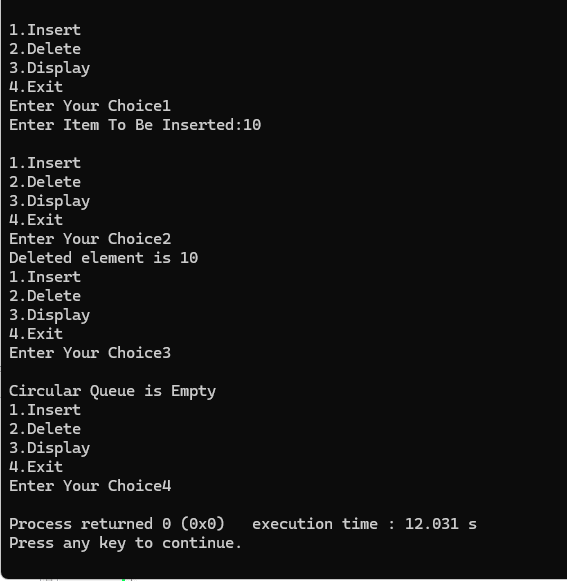
**printf("%d\t",cq[i]);**

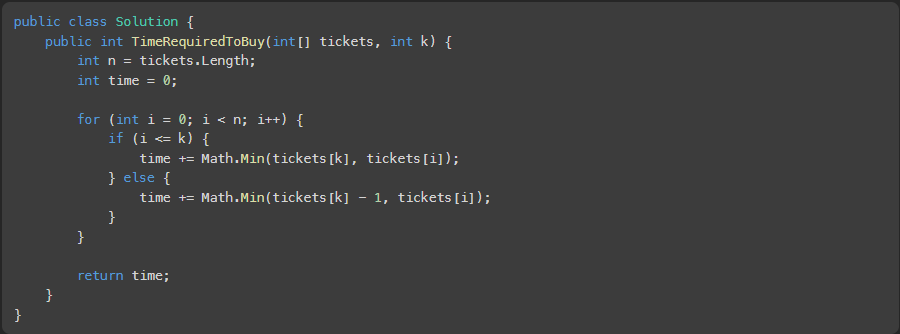
**}**

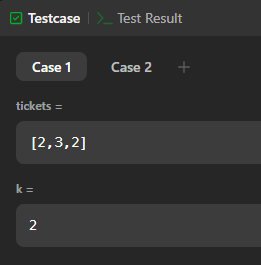
**}**

**return;**

**}**







5) Programs for :

1) Factorial using recursion   
2) Fibonacci using recursion   
3) Tower of Hanoi using recursion   
4) Leet Code challenge

#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:");

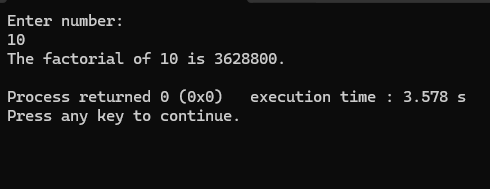
scanf("%d",&n);

t=n;

f1=fact(n);

printf("Factorial of %d is %d",t,f1);

}



#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;

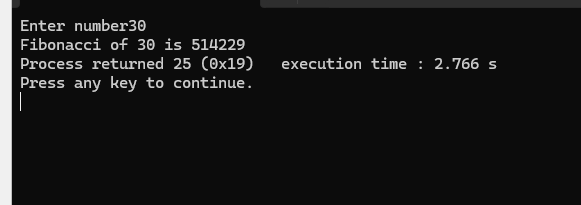
printf("Enter number:");

scanf("%d",&n);

fib=fibo(n);

printf("Fibonacci number:%d",fib);

}



#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;

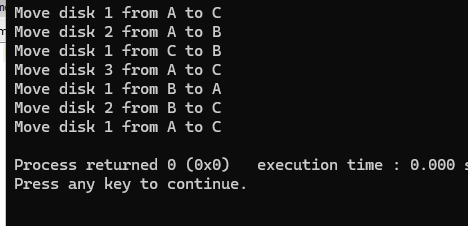
char s='S',d='D',t='T';

printf("Enter number:");

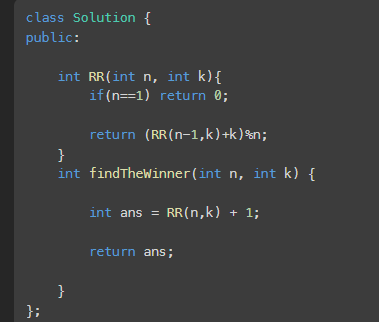
scanf("%d",&n);

toh(n,s,t,d);

}



Leetcode:

v

6)

A) 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.

#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");

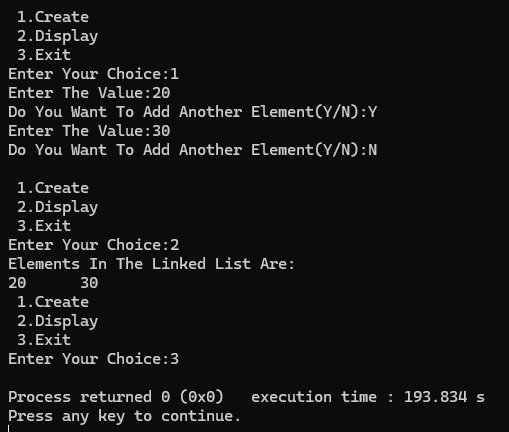
return;

}

new1->link=temp->link;

temp->link=new1;

}



B) 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.

#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;

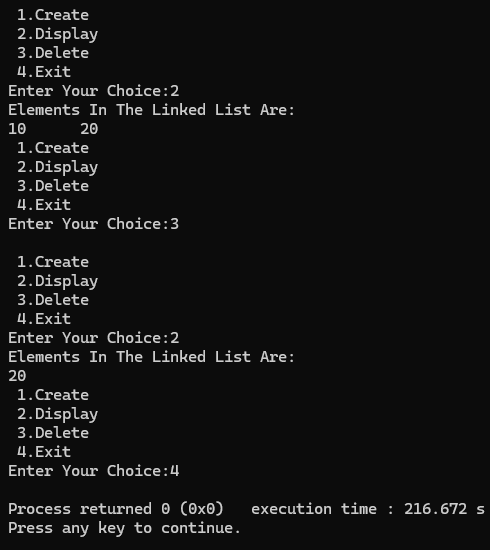
}

else{

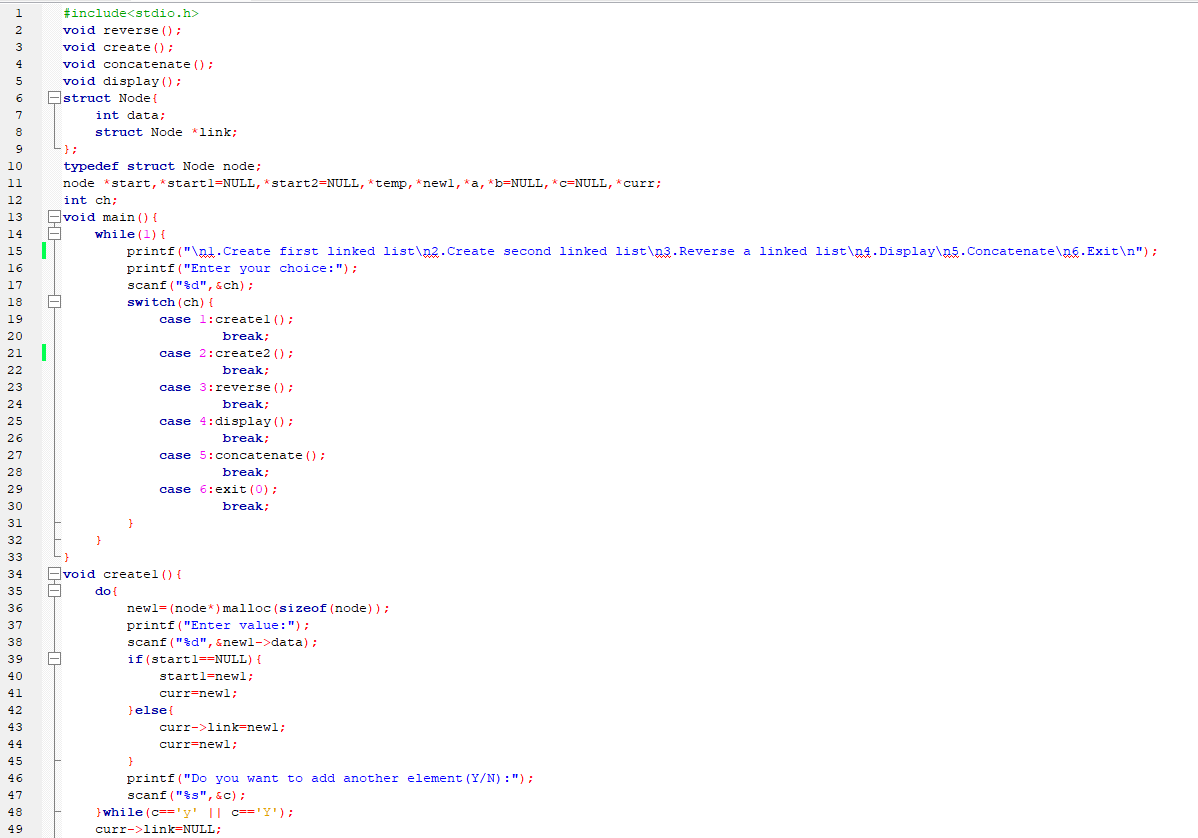
printf("Element not found");

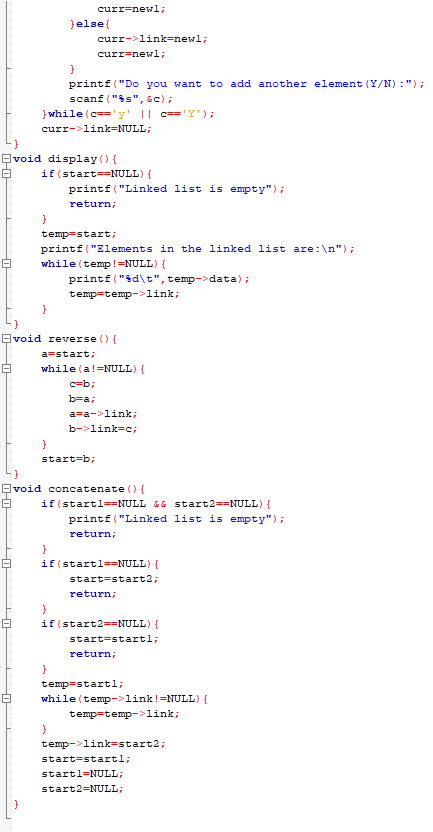
}

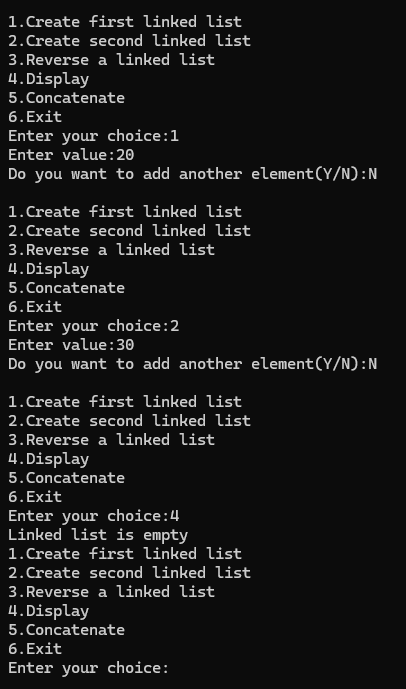
}



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

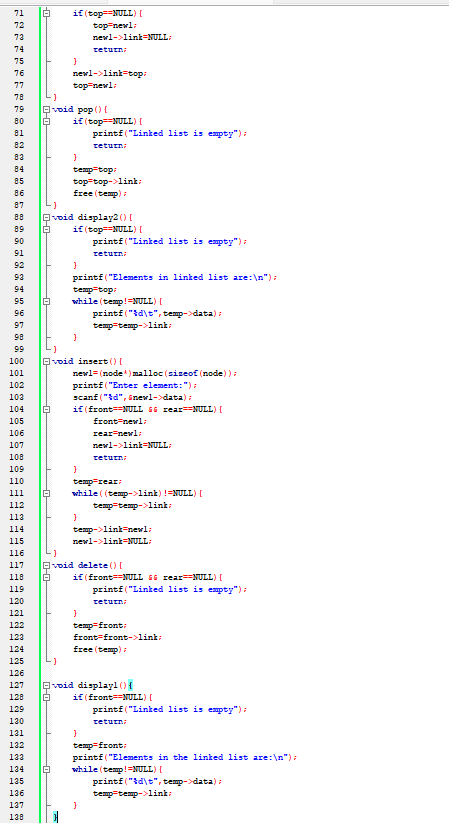


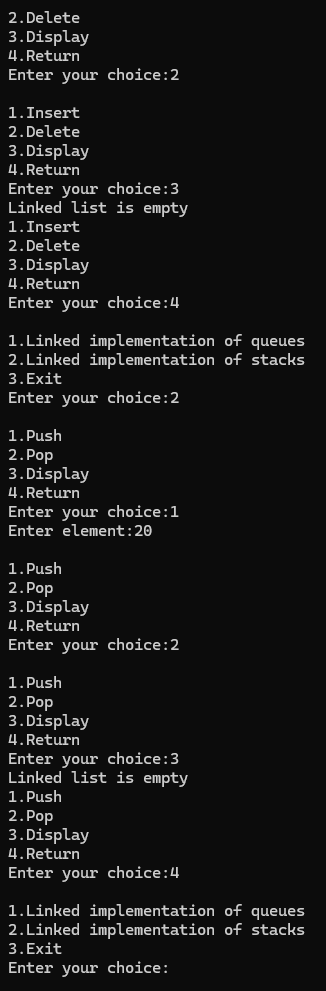




7) b) WAP to Implement Single Link List to simulate Stack & Queue Operations.







8) Implementing single linked list to simulate stack and queue operations.

#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.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:2

1.Insert

2.Delete

3.Display

4.Return

Enter your choice:3

Linked list is empty

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:2

1.Push

2.Pop

3.Display

4.Return

Enter your choice:3

Linked list is empty

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 \*/

9) Doubly linked list.

#include<stdio.h>

struct Node{

struct Node \*pre;

int data;

struct Node \*next;

};

typedef struct Node node;

node \*new1,\*curr,\*temp,\*start=NULL,\*new2;

void create();

void insert\_beg();

void insert\_end();

void insert\_at\_req\_position();

void delete\_first();

void delete\_last();

void delete\_specific\_element();

void display();

int ch,pos,ele;

char c;

void main(){

while(1){

printf("\n1.Create DLL\n2.Insert at beginning\n3.Insert at last\n4.Insert at required position\n5.Delete first element\n6.Delete last element\n7.Delete specific element\n8.Display\n9.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\_req\_position();

break;

case 5:delete\_first();

break;

case 6:delete\_last();

break;

case 7:delete\_specific\_element();

break;

case 8:display();

break;

case 9:exit(0);

break;

}

}

}

void create(){

new1=(node\*)malloc(sizeof(node));

printf("Enter the data:");

scanf("%d",&new1->data);

new1->pre=NULL;

start=new1;

curr=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 element:");

scanf("%d",&new1->data);

new1->pre=curr;

curr->next=new1;

curr=new1;

}

else{

curr->next=NULL;

return;

}

}

}

void insert\_beg(){

new1=(node\*)malloc(sizeof(node));

printf("Enter element:");

scanf("%d",&new1->data);

if(start==NULL){

new1->pre=NULL;

new1->next=NULL;

start=new1;

return;

}

new1->pre=NULL;

new1->next=start;

start->pre=new1;

start=new1;

}

void insert\_end(){

new1=(node\*)malloc(sizeof(node));

printf("Enter element:");

scanf("%d",&new1->data);

if(start==NULL){

new1->pre=NULL;

new1->next=NULL;

start=new1;

return;

}

temp=start;

while(temp->next!=NULL){

temp=temp->next;

}

temp->next=new1;

new1->pre=temp;

new1->next=NULL;

}

void insert\_at\_req\_position(){

new1=(node\*)malloc(sizeof(node));

printf("Enter element:");

scanf("%d",&new1->data);

printf("Enter position:");

scanf("%d",&pos);

if(pos==1){

new1->pre=NULL;

new1->next=start;

start->pre=new1;

start=new1;

return;

}

int i=1;

temp=start;

while(i<pos-1 && temp!=NULL){

temp=temp->next;

i++;

}

if(i==pos-1){

new1->next=temp->next;

temp->next->pre=new1;

new1->pre=temp;

temp->next=new1;

return;

}

if(temp==NULL){

printf("Entered position is greater than number of elements");

}

}

void delete\_first(){

if(start==NULL){

printf("Doubly linked list is empty");

return;

}

temp=start;

start=start->next;

start->pre=NULL;

free(temp);

}

void delete\_last(){

if(start==NULL){

printf("Doubly linked list is empty");

return;

}

temp=start;

while(temp->next!=NULL){

temp=temp->next;

}

temp->pre->next=NULL;

free(temp);

}

void delete\_specific\_element(){

if(start==NULL){

printf("Doubly linked list is empty");

return;

}

printf("Enter element to be deleted:");

scanf("%d",&ele);

if(start->data==ele){

temp=start;

start=start->next;

start->pre=NULL;

free(temp);

return;

}

temp=start;

while(temp->data!=ele && temp->next!=NULL){

temp=temp->next;

}

if(temp->data==ele && temp->next==NULL){

temp->pre->next=NULL;

free(temp);

return;

}

if(temp->data==ele){

temp->pre->next=temp->next;

temp->next->pre=temp->pre;

free(temp);

return;

}

if(temp==NULL){

printf("Element not found");

}

}

void display(){

if(start==NULL){

printf("Doubly linked list is empty");

return;

}

temp=start;

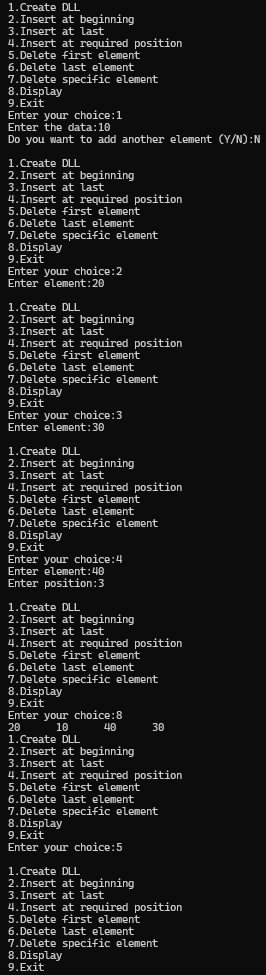
while(temp!=NULL){

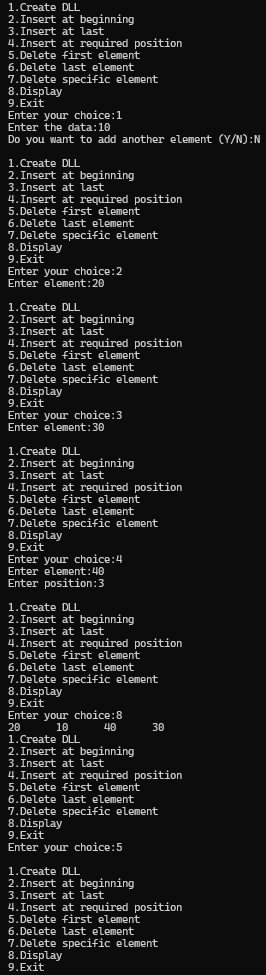
printf("%d\t",temp->data);

temp=temp->next;

}

}





9) Implementation of Binary Search.

#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){

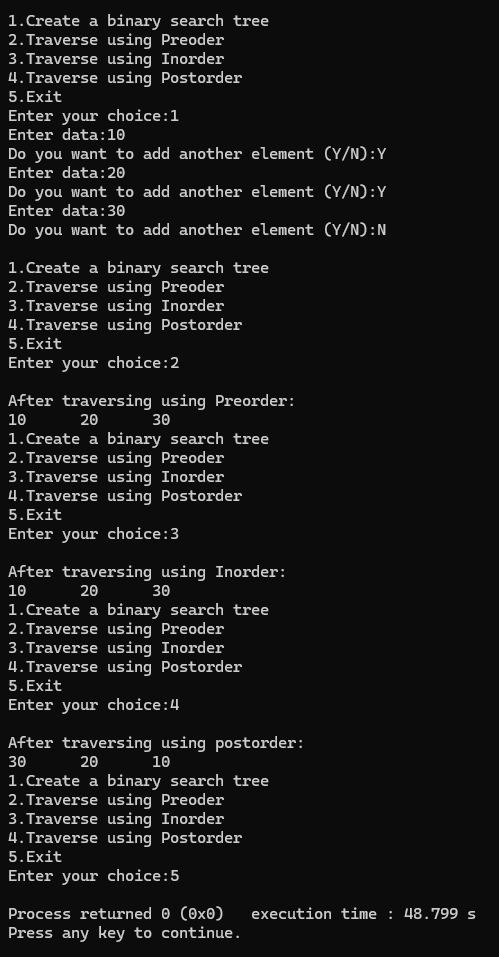
postorder(temp->left);

postorder(temp->right);

printf("%d\t",temp->data);

}

}



10) A) PROGRAM FOR DFS TRAVERSAL.

#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++){

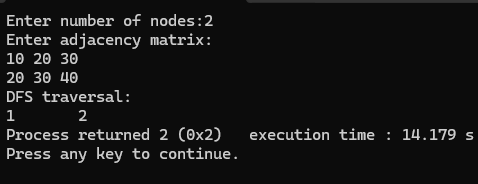
if(vis[k]==0 && a[v][k]==1){

dfs(k);

}

}

}



10) B) PROGRAM TO IMPLEMENT BFS TRAVERSAL.

#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;

}

}

}

}

