B.M.S COLLEGE OF ENGINEERING

DATA STUCTURE LAB RECORD

NAME : ANVITHA ARAVINDA

USN : 1BM19CS021

SUB : DS/LAB

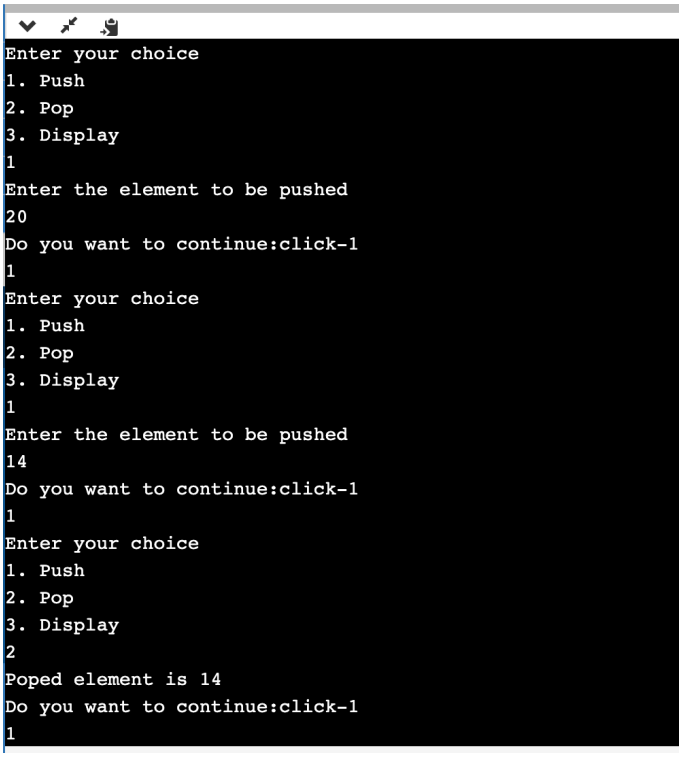
ACADEMIC YEAR : 2020-2021

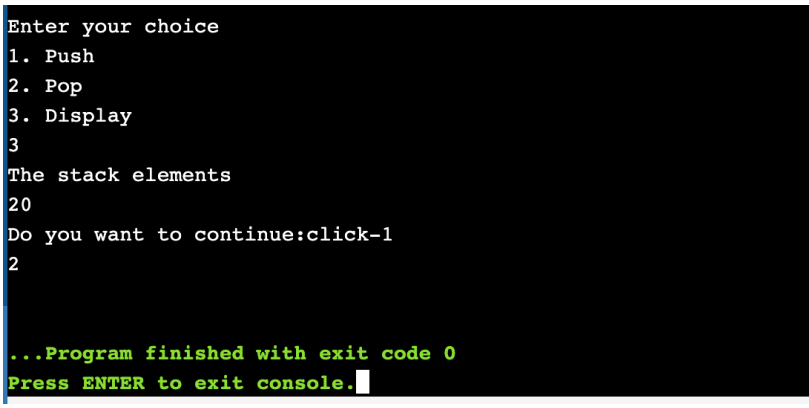
PROGRAM 1:

```
#include <stdio.h>
#define size 5
int top=-1;
void push(int [], int);
int pop(int[]);
void display(int []);
int main()
{
int a[5];

int choice,element;
int ch;
do
{
printf("Enter your choice\n");
printf("1. Push\n");
printf("2. Pop\n");
printf("3. Display\n");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("Enter the element to be pushed \n");
scanf("%d",&element);
push(a,element);
break;
case 2: element=pop(a);
if(element==-1)
printf("Stack Underflow cannot remove\n");
else
printf("Poped element is %d \n",element);
break;
```

```
case 3: display(a);
break;
default: printf("Invalid choice\n");
}
printf("Do you want to continue:click-1\n");
scanf("%d",&ch);
} while(ch==1);
return 0;
}

void push(int a[], int ele)
{
if (top==size-1)
{
printf("Stack overflow cannot push\n");
}
else
{
top++;
a[top]=ele;
}
}

int pop(int a[])
{
int ele;
if(top==-1)

return -1;

else
{
ele=a[top];
top--;
return (ele);
}
```

```
}

void display(int a[])
{
int i;
printf("The stack elements\n");
for(i=top;i>=0;i--)
{

printf("%d\t",a[i]);
}
printf("\n");
}
```





PROGRAM 2:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define SIZE 100


char stack[SIZE];
int top=-1;

void push(char ch)
{
if (top==SIZE-1)
printf("STACK OVERFLOW!! Stack is full!!\n");
else
{
top++;
stack[top]=ch;
}
}

char pop()
{
char item;
if (top==-1)
printf("\nSTACK UNDERFLOW!! Stack is empty!!");
else
{
item=stack[top];
top--;
return item;
}
}
```

```
int stackempty()
{
if(top==-1)
return 1;
else
return 0;
}

char stacktop()
{
if( top==-1)
printf("\nSTACK UNDERFLOW!! Stack is empty!!");
else
return stack[top];
}

int priority(char ch)
{
switch(ch)
{
case '+':
case '-':return (1);
case '*':
case '/': return (2);
case '^': return (3);
default : return (0);
}
}

int main(int argc, char **argv)
{
char infix[100];
int i, item;
printf("Enter a valid infix expression : ");
scanf("%s",infix);
```

```
printf("\n----------------------------------------
-----\n");
for (int i = 0; i < strlen(infix); ++i)
{
if((infix[i]=='*' || infix[i]=='+' ||
infix[i]=='/' ||
infix[i]=='-' || infix[i]=='^' || infix[i]=='(')
&&
(infix[i+1]=='*' || infix[i+1]=='+' ||
infix[i+1]=='/' ||
infix[i+1]=='-' || infix[i+1]=='^' ||
infix[i+1]==')'))
{
printf("INVALID EXPRESSION");
exit(1);
}
}
printf("The entered Infix Expression is :
%s",infix);
printf("\nThe generated Postfix Expression is :
");
i=0;
while (infix[i]!='\0')
{
switch (infix[i])
{
case '(': push(infix[i]);
break;
case ')': while(( item=pop())!='(')
printf("%c",item);
break;
case '+':
case '-':
case '*':
case '/':
case '^':
while(!stackempty() &&
priority(infix[i])<=priority(stacktop()))
```

```
{
item=pop();

printf("%c", item);
}

push(infix[i]);
break;
default : printf("%c", infix[i]);
break;
}
i++;
}

while(!stackempty())
{
char item;
item=pop();
printf("%c", item);

}
printf("\n");
return 0;
}
```

```
Enter a valid infix expression : a+b*(c^d-e)^(f+g*h)-i

-----------------------------------------
The entered Infix Expression is : a+b*(c^d-e)^(f+g*h)-i
The generated Postfix Expression is : abcd^e-fgh*+^*+i-
```

PROGRAM 3:

```
#include<stdio.h>
#include<stdlib.h>
#define max 5
int insertq ( int queue[max], int *rear , int *data,int *front)
{
if ( *rear == max -1 )
return(-1);
else
{     if(*front==-1)
*front=0;
*rear = *rear + 1;
queue[*rear] = *data;
return(1);
}
}
int delq( int queue[max], int *front, int *rear)
{

if ( *front == *rear)
return(-1);
else
{


printf("deleted:%d",queue[*front]);
(*front)++;
if(*front==*rear)
*front=*rear=-1;
return(1);
}
}
```

```
void display(int queue[max],int *front,int
*rear)
{
int i;
if(*rear==-1)
printf("Queue is empty\n");
else
{
printf("\n Queue contents:");
for(i=*front;i<=*rear;i++)
printf("%d", queue[i]);
}
}
int main()
{
int queue[max],data,i;
int front,rear,reply,option;
front =-1;
rear = -1;
printf("\tMenu");
printf("\n-------------------------");
printf("\n 1. Insert element in queue");
printf("\n 2. Delete element from queue");
printf("\n 3.Display");
printf("\n 4. Exit");
printf("\n-------------------------");
do
{
printf("\nChoose operation : ");
scanf("%d",&option);
switch(option)
{
case 1 :
printf("\nEnter Number : ");
scanf("%d",&data);
reply = insertq(queue,&rear,&data,&front);
```

```
                if ( reply == - 1)
                printf("Queue is full");
                break;
                case 2 :
                reply = delq(queue,&front,&rear);
                if ( reply == -1 )
                printf("Queue is empty ");
                break;
                case 3:
                display(queue,&front,&rear);
                break;

                case 4 : exit(0);
                }
                }while(option!=4);
                }
```

```
        Menu
----------------------------
 1. Insert element in queue
 2. Delete element from queue
 3.Display
 4. Exit
----------------------------
Choose operation : 1

Enter Number : 12

Choose operation : 1

Enter Number : 24

Choose operation : 3

 Queue contents:12      24
Choose operation : 2
deleted:12
Choose operation : 3
Queue is empty

Choose operation : 4
```

PROGRAM 4:

```
#include <stdio.h>
#include <stdlib.h>
#define S 3

int front=-1;
int rear=-1;

int queue[S];

void Enque(int, int);
int Deque(int);
void display(int);
int main(int argc, char **argv)
{
int choice, SIZE;
int item;
printf("Enter the SIZE of the queue : \n");
scanf("%d",&SIZE);
do
{
printf("\n<----------CIRCULAR QUEUE---------->\n");
printf("\n 1. INSERT to Queue (EnQueue)");
printf("\n 2. DELETE from the Queue (DeQueue)");
printf("\n 3. DISPLAY the content ");
printf("\n 4. EXIT\n");
printf("\n<---------------------------------->\n");
printf("Enter your choice accordingly : ");
scanf("%d", &choice);
switch(choice)
{
case 1:  if(((front == 0 && rear == SIZE - 1)) || (front == rear + 1))
```

```
{
printf("-----Queue is FULL-----\n");
break;
}

printf("\nEnter the element you want to INSERT : \n");
scanf("%d",&item);
Enque(SIZE, item);
break;

case 2: item=Deque(SIZE);
if(item==-999)
printf("-----Queue is EMPTY-----\n");
else
printf("\nRemoved element from the queue %d\n",item);
break;

case 3: display(SIZE);
break;

case 4: printf("EXITING......\n");
exit(0);
default: printf("INVALID CHOICE!!");
break;
}
}
while (choice!=4);
return 0;
}

void Enque(int SIZE, int ele)
{
if(((front == 0 && rear == SIZE - 1)) || (front == rear + 1) )
{
```

```
printf("-----Queue is FULL-----\n");
return;
}
else
{
rear=(rear+1)%SIZE;
queue[rear]=ele;
if(front == -1)
front=0;
}
}

int Deque(int SIZE)
{
int item;
if((front == -1)&&(rear == -1))
{
return(-999);
}
else
{
item=queue[front];

if(front==rear)
{
front=-1;
rear=-1;
}
else
{
front=(front+1)%SIZE;
}
return item;
}

}
```

```
void display(int SIZE)
{
int i;
if(((front==-1)&& (rear==-1)))
{
printf("-----Queue is EMPTY-----\n");
return;
}
else
{
printf("\nQueue contents:\n");
for(i=front;i!=rear;i=(i+1)%SIZE)
{
printf("%d\t", queue[i]);
}
printf("%d\t", queue[i]);
}
}
```

```
Enter the SIZE of the queue :
4

<----------CIRCULAR QUEUE---------->

 1. INSERT to Queue (EnQueue)
 2. DELETE from the Queue (DeQueue)
 3. DISPLAY the content
 4. EXIT

<---------------------------------->
Enter your choice accordingly : 1

Enter the element you want to INSERT :
12

<----------CIRCULAR QUEUE---------->

 1. INSERT to Queue (EnQueue)
 2. DELETE from the Queue (DeQueue)
 3. DISPLAY the content
 4. EXIT

<---------------------------------->
Enter your choice accordingly : 1
```

```
Enter the element you want to INSERT :
23

<----------CIRCULAR QUEUE---------->

 1. INSERT to Queue (EnQueue)
 2. DELETE from the Queue (DeQueue)
 3. DISPLAY the content
 4. EXIT

<---------------------------------->
Enter your choice accordingly : 3

Queue contents:
12      23
<----------CIRCULAR QUEUE---------->

 1. INSERT to Queue (EnQueue)
 2. DELETE from the Queue (DeQueue)
 3. DISPLAY the content
 4. EXIT

<---------------------------------->
Enter your choice accordingly : 2
```

```
Removed element from the queue 12

<----------CIRCULAR QUEUE---------->

 1. INSERT to Queue (EnQueue)
 2. DELETE from the Queue (DeQueue)
 3. DISPLAY the content
 4. EXIT

<---------------------------------->
Enter your choice accordingly : 3

Queue contents:
23
<----------CIRCULAR QUEUE---------->

 1. INSERT to Queue (EnQueue)
 2. DELETE from the Queue (DeQueue)
 3. DISPLAY the content
 4. EXIT

<----------------------------------
```

PROGRAM 5:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node
{
int sem;
char name[50];
char usn[50];
struct node *next;
};
struct node *head= NULL;
int c=0;
void Insertbegining()
{
struct node *newnode;
int s;
char a[50],b[50];
printf("Enter your name  : ");
scanf("%s",a);
printf("Enter your usn  : ");
scanf("%s",b);
printf("Enter your semester  : ");
scanf("%d",&s);

newnode=(struct node*)malloc(sizeof(struct node));
newnode->sem =s;
strcpy(newnode->name,a);
strcpy(newnode->usn,b);

newnode->next=head;
```

```
head=newnode;
c++;
printf("Node created\n");
}
void Insertany(int p)
{
struct node *newnode;
int s;
char a[30],b[30];
printf("Enter your name  : ");
scanf("%s",a);
printf("Enter your usn  : ");
scanf("%s",b);
printf("Enter your semester  : ");
scanf("%d",&s);

newnode=(struct node*)malloc(sizeof(struct node));
newnode->sem =s;
strcpy(newnode->name,a);
strcpy(newnode->usn,b);
if(p==1)
{
printf("Node of linked list is inserted in the first position\n");
newnode->next=head;
head=newnode;
c++;
}
else if(head==NULL && p>1)
{
printf("the list is empty and node cannot be created\n");
return;
}
else if(p>(c+1))
{
```

```
printf("Not possible since number of nodes existing in the list is insufficient\n");
return;
}
else
{
struct node *temp1;
struct node *temp2;
int count=1;
temp1=head;
while(count<(p-1))
{
temp1= temp1->next;
count++;
}
temp2= temp1->next;
temp1->next=newnode;
newnode->next=temp2;
c++;
printf("Node inserted at %d position in linked list\n",p);
}
}

void Insertend()
{
struct node *newnode;
struct node *temp;
int s;
char n[30],u[30];
printf("Enter your name  : ");
scanf("%s",n);
printf("Enter your semester  : ");
scanf("%d",&s);
printf("Enter your usn  : ");
scanf("%s",u);
```

```
newnode=(struct node*)malloc(sizeof(struct node));
newnode->sem =s;
strcpy(newnode->name,n);
strcpy(newnode->usn,u);
if (head==NULL)
{
newnode->next=NULL;
head=newnode;
printf("first node of linked list created\n");
c++;
}
else
{
temp=head;
while(temp->next!=NULL)
{
temp=temp->next;
}
temp->next=newnode;
newnode->next=NULL;
c++;
printf("Node created\n");
}
}
void display()
{
struct node *ptr;
ptr=head;
int i=1;

if(ptr==NULL)
{
printf("Linked list is empty!\n");
}
else
{
```

```
while(ptr!= NULL)
{
printf("----NODE %d----\n",i);
printf("Name: %s\n",ptr->name);
printf("USN: %s\n",ptr->usn);
printf("Sem: %d\n",ptr->sem);
printf("\n");
i++;
ptr=ptr->next;
}

}

}


int main()
{
int choice,pos;
do
{

printf("\n1. Insert node at beginning of the list\n2. Insert node anywhere in the list\n3. Insert at the end of list\n4. Display list\n5. Exit\n");
printf("\nEnter your choice : ");
scanf("%d",&choice);
if(choice==5)
break;
switch(choice)
{
case 1:
Insertbegining();
break;

case 2:
```

```
                printf("Enter in which position of the list you want to
                enter your node\n");
                scanf("%d",&pos);
                Insertany(pos);
                break;

                case 3:
                Insertend();
                break;

                case 4:
                display();
                break;

                default:
                printf("Wrong choice!\n");
                break;
                }
                }while(choice!=5);
                return 0;
                }
```

```
1. Insert node at beginning of the list
2. Insert node anywhere in the list
3. Insert at the end of list
4. Display list
5. Exit

Enter your choice : 1
Enter your name  : ANVI
Enter your usn  : 1BM19CS021
Enter your semester  : 3
Node created

1. Insert node at beginning of the list
2. Insert node anywhere in the list
3. Insert at the end of list
4. Display list
5. Exit

Enter your choice : 3
Enter your name  : AAA
Enter your semester  : 1BM19CS001
Enter your usn  : Node created
```

```
1. Insert node at beginning of the list
2. Insert node anywhere in the list
3. Insert at the end of list
4. Display list
5. Exit

Enter your choice : 4
----NODE 1----
Name: ANVI
USN: 1BM19CS021
Sem: 3

----NODE 2----
Name: AAA
USN: BM19CS001
Sem: 1


1. Insert node at beginning of the list
2. Insert node anywhere in the list
3. Insert at the end of list
4. Display list
5. Exit
```

```
Enter your choice : 2
Enter in which position of the list you want to enter your node
2
Enter your name  : ABC
Enter your usn  : 1BM19CS003
Enter your semester  : 3
Node inserted at 2 position in linked list

1. Insert node at beginning of the list
2. Insert node anywhere in the list
3. Insert at the end of list
4. Display list
5. Exit

Enter your choice : 4
----NODE 1----
Name: ANVI
USN: 1BM19CS021
Sem: 3
```

```
----NODE 2----
Name: ABC
USN: 1BM19CS003
Sem: 3

----NODE 3----
Name: AAA
USN: BM19CS001
Sem: 1


1. Insert node at beginning of the list
2. Insert node anywhere in the list
3. Insert at the end of list
4. Display list
5. Exit

Enter your choice : 5
```

PROGRAM 6:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node
{
int sem;
char name[50];
char usn[50];
struct node *next;
};
struct node *head= NULL;
int c=0;
void Insert()
{
struct node *newnode;
struct node *temp;
int s;
char n[30],u[30];
printf("Enter your name  : ");
scanf("%s",n);
printf("Enter your semester  : ");
scanf("%d",&s);
printf("Enter your usn  : ");
scanf("%s",u);
newnode=(struct node*)malloc(sizeof(struct node));
newnode->sem =s;
strcpy(newnode->name,n);
strcpy(newnode->usn,u);
if (head==NULL)
{
newnode->next=NULL;
```

```
head=newnode;
printf("first node of linked list created\n");
c++;
}
else
{
temp=head;
while(temp->next!=NULL)
{
temp=temp->next;
}
temp->next=newnode;
newnode->next=NULL;
c++;
printf("Node created\n");
}
}
void deletebeg()
{
struct node *ptr;
if(head == NULL)
{
printf("\nList is empty");
}
else
{
ptr = head;
head = ptr->next;
free(ptr);
printf("\n Node deleted from the begining ...");
}
}
void deletemid()
{
char key[20];
printf("enter the usn of student to be deleted\n");
```

```
scanf("%s",key);
struct node *temp = head, *prev;
if (temp != NULL && strcmp(temp->usn,key)==0)
{
head = temp->next;
free(temp);
return;
}
while (temp != NULL && strcmp(temp->usn,key)!=0)
{
prev = temp;
temp = temp->next;
}

if (temp == NULL)
{
printf("student not in the list\n");
return;
}
prev->next = temp->next;

free(temp);
}
void deleteend()
{

struct node *toDelLast, *preNode;
if(head == NULL)
{
printf(" There is no element in the list.");
}
else
{
toDelLast = head;
preNode = head;
while(toDelLast->next != NULL)
```

```
{
preNode = toDelLast;
toDelLast = toDelLast->next;
}
if(toDelLast == head)
{
head = NULL;
}
else
{
preNode->next = NULL;
}
free(toDelLast);
}
}

void display()
{
struct node *ptr;
ptr=head;
int i=1;

if(ptr==NULL)
{
printf("Linked list is empty!\n");
}
else
{
while(ptr!= NULL)
{
printf("----NODE %d----\n",i);
printf("Name: %s\n",ptr->name);
printf("USN: %s\n",ptr->usn);
printf("Sem: %d\n",ptr->sem);
printf("\n");
i++;
```

```
ptr=ptr->next;
}

}

}

int main()
{
int choice,pos;
do
{

printf("\n1. Insert node \n2. delete node in the beg of the list\n3. delete at the end of list\n4.delete a given node \n5. display list\n6.exit\n");
printf("\nEnter your choice : ");
scanf("%d",&choice);
switch(choice)
{
case 1:
Insert();
break;

case 2:
deletebeg();
break;

case 3:
deleteend();
break;

case 4:
deletemid();
break;
```

```
case 5:
display();
break;

case 6:
break;

default:
printf("Wrong choice!\n");
break;
}
}while(choice!=6);
return 0;
}
```

```
1. Insert node
2. delete node in the beg of the list
3. delete at the end of list
4.delete a given node
5. display list
6.exit

Enter your choice : 1
Enter your name  : aaa
Enter your semester  : 3
Enter your usn  : 12345
first node of linked list created

1. Insert node
2. delete node in the beg of the list
3. delete at the end of list
4.delete a given node
5. display list
6.exit
```

```
Enter your choice : 1
Enter your name  : ssss
Enter your semester  : 3
Enter your usn  : 23456
Node created

1. Insert node
2. delete node in the beg of the list
3. delete at the end of list
4.delete a given node
5. display list
6.exit

Enter your choice : 1
Enter your name  : dddd
Enter your semester  : 3
Enter your usn  : 45678
Node created
```

```
1. Insert node
2. delete node in the beg of the list
3. delete at the end of list
4.delete a given node
5. display list
6.exit

Enter your choice : 5
----NODE 1----
Name: aaa
USN: 12345
Sem: 3

----NODE 2----
Name: ssss
USN: 23456
Sem: 3

----NODE 3----
Name: dddd
USN: 45678
Sem: 3
```

```
1. Insert node
2. delete node in the beg of the list
3. delete at the end of list
4.delete a given node
5. display list
6.exit

Enter your choice : 4
enter the usn of student to be deleted
23456

1. Insert node
2. delete node in the beg of the list
3. delete at the end of list
4.delete a given node
5. display list
6.exit

Enter your choice : 5
```

```
Enter your choice : 5
----NODE 1----
Name: aaa
USN: 12345
Sem: 3

----NODE 2----
Name: dddd
USN: 45678
Sem: 3


1. Insert node
2. delete node in the beg of the list
3. delete at the end of list
4.delete a given node
5. display list
6.exit

Enter your choice : 2

 Node deleted from the begining ...
```

```
1. Insert node
2. delete node in the beg of the list
3. delete at the end of list
4.delete a given node
5. display list
6.exit

Enter your choice : 5
----NODE 1----
Name: dddd
USN: 45678
Sem: 3


1. Insert node
2. delete node in the beg of the list
3. delete at the end of list
4.delete a given node
5. display list
6.exit

Enter your choice : 3
```

```
Enter your choice : 3

1. Insert node
2. delete node in the beg of the list
3. delete at the end of list
4.delete a given node
5. display list
6.exit

Enter your choice : 5
Linked list is empty!

1. Insert node
2. delete node in the beg of the list
3. delete at the end of list
4.delete a given node
5. display list
6.exit

Enter your choice : 6


...Program finished with exit code 0
```

PROGRAM 7:

```
#include <stdlib.h>
#include <string.h>
struct node
{
int sem;
struct node *next;
};
struct node *head= NULL;
struct node *head2= NULL;
int c=0;
void Insert()
{
struct node *newnode;
struct node *temp;
int s;
printf("Enter integer  : ");
scanf("%d",&s);
newnode=(struct node*)malloc(sizeof(struct node));
newnode->sem =s;
if (head==NULL)
{
newnode->next=NULL;
head=newnode;
printf("first node of linked list created\n");
c++;
}
else
{
temp=head;
while(temp->next!=NULL)
{
temp=temp->next;
```

```
}
temp->next=newnode;
newnode->next=NULL;
c++;
printf("Node created\n");
}
}
void Insert2()
{
struct node *newnode;
struct node *temp;
int s,y;
printf("enter elements to create list 2\n");
do
{
printf("Enter integer  : \n");
scanf("%d",&s);
newnode=(struct node*)malloc(sizeof(struct node));
newnode->sem =s;
if (head2==NULL)
{
newnode->next=NULL;
head2=newnode;
printf("first node of linked list created\n");
c++;
}
else
{
temp=head2;
while(temp->next!=NULL)
{
temp=temp->next;
}
temp->next=newnode;
newnode->next=NULL;
c++;
```

```
printf("Node created\n");
}
printf("do u want to continue adding:0 or 1\n");
scanf("%d",&y);
}while(y!=0);
}


void bubbleSort()
{
int swapped, i;
struct node *ptr1;
struct node *lptr = NULL;


if (head == NULL)
return;

do
{
swapped = 0;
ptr1 = head;

while (ptr1->next != lptr)
{
if (ptr1->sem > ptr1->next->sem)
{
int temp = ptr1->sem;
ptr1->sem = ptr1->next->sem;
ptr1->next->sem = temp;
swapped = 1;
}
ptr1 = ptr1->next;
}
lptr = ptr1;
}
```

```
while (swapped);
}

void reverse()
{
struct node* prev = NULL;
struct node* current = head;
struct node* next = NULL;
while (current != NULL) {
next = current->next;
current->next = prev;
prev = current;
current = next;
}
head= prev;
}

void concat()
{
struct node *ptr;
if(head==NULL)
{
head=head2;
}
if(head2==NULL)
{
head2=head;
}
ptr=head;
while(ptr->next!=NULL)
ptr=ptr->next;
ptr->next=head2;
}
void display1()
{
struct node *ptr;
```

```
ptr=head;
int i=1;

if(ptr==NULL)
{
printf("Linked list is empty!\n");
}
else
{
while(ptr!= NULL)
{
printf(" %d",ptr->sem);
i++;
ptr=ptr->next;
}

}

}
void display2()
{
struct node *ptr;
ptr=head2;
int i=1;

if(ptr==NULL)
{
printf("Linked list is empty!\n");
}
else
{
while(ptr!= NULL)
{

printf(" %d",ptr->sem);
```

```
printf("\n");
i++;
ptr=ptr->next;
}

}

}

int main()
{
int choice,pos;
do
{

printf("\n1. Insert node \n2. sort node\n3. reverse node\n4.concat 2 lists \n5.exit\n");
printf("\nEnter your choice : ");
scanf("%d",&choice);
switch(choice)
{
case 1:
Insert();
break;

case 2:
printf("before:\n");
display1();
bubbleSort();
printf("after:\n");
display1();
break;

case 3:
printf("before:\n");
display1();
```

```
                reverse();
                printf("after:\n");
                
                display1();
                break;
                
                case 4:
                Insert2();
                concat();
                display1();
                break;
                
                case 5:
                break;
                
                default:
                printf("Wrong choice!\n");
                break;
                }
                }while(choice!=5);
                return 0;
                }
```

```
1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit

Enter your choice : 1
Enter integer  : 12
first node of linked list created

1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit

Enter your choice : 1
Enter integer  : 13
Node created
```

```
1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit

Enter your choice : 1
Enter integer  : 45
Node created

1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit

Enter your choice : 1
Enter integer  : 2
Node created
```

```
1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit

Enter your choice : 3
before:
 2 3 12 13 45after:
 45 13 12 3 2
1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit

Enter your choice : 4
enter elements to create list 2
Enter integer  :
12
first node of linked list created
do u want to continue adding:0 or 1
0
```

```
1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit

Enter your choice : 1
Enter integer  : 3
Node created

1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit

Enter your choice : 2
before:
 12 13 45 2 3after:
 2 3 12 13 45
```

```
3. reverse node
4.concat 2 lists
5.exit

Enter your choice : 4
enter elements to create list 2
Enter integer  :
12
first node of linked list created
do u want to continue adding:0 or 1
0
 45 13 12 3 2 12
1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit

Enter your choice : 5


...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM 8:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int info;
struct node *ptr;
}*top,*top1,*temp;

void push(int data);
void pop();
void display();
void create();


int main()
{
int no, ch, e;

printf("\n 1 - Push");
printf("\n 2 - Pop");
printf("\n 3 - Dipslay");
printf("\n 4 - Exit");

create();

while (1)
{
printf("\n Enter choice : ");
scanf("%d", &ch);

switch (ch)
{
```

```
case 1:
printf("Enter data : ");
scanf("%d", &no);
push(no);
break;
case 2:
pop();
break;
case 3:
display();
break;
case 4:
exit(0);
default :
printf(" Wrong choice, Please enter correct choice  ");
}
}
}


void create()
{
top = NULL;
}


void push(int data)
{
if (top == NULL)
{
top =(struct node *)malloc(1*sizeof(struct node));
top->ptr = NULL;
top->info = data;
}
```

```
else
{
temp =(struct node *)malloc(1*sizeof(struct node));
temp->ptr = top;
temp->info = data;
top = temp;
}

}


void display()
{
top1 = top;

if (top1 == NULL)
{
printf("Stack is empty");
return;
}

while (top1 != NULL)
{
printf("%d ", top1->info);
top1 = top1->ptr;
}
}

void pop()
{
top1 = top;

if (top1 == NULL)
{
```

```
printf("\n Error : Trying to pop from empty stack");
return;
}
else
top1 = top1->ptr;
printf("\n Popped value : %d", top->info);
free(top);
top = top1;

}
```


input
1 - Push
2 - Pop
3 - Dipslay
4 - Exit
Enter choice : 1
Enter data : 12
Enter choice : 1
Enter data : 23
Enter choice : 2
Popped value : 23
Enter choice : 3
12
Enter choice : 4
..Program finished with exit code 0
Press ENTER to exit console.


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

struct node
{
int info;
struct node *ptr;
}*front,*rear,*temp,*front1;


void enq(int data);
void deq();

void display();
void create();


int main()
{
int no, ch, e;

printf("\n 1 - Enque");
printf("\n 2 - Deque");
printf("\n 3 - Display");
printf("\n 4 - Exit");
create();
while (1)
{
printf("\n Enter choice : ");
scanf("%d", &ch);
switch (ch)
{
case 1:
```

```
printf("Enter data : ");
scanf("%d", &no);
enq(no);
break;
case 2:
deq();
break;
case 3:
display();
break;
case 4:
exit(0);
default:
printf("Wrong choice, Please enter correct choice  ");
break;
}
}
return 0;
}


void create()
{
front = rear = NULL;
}


void enq(int data)
{
if (rear == NULL)
{
rear = (struct node *)malloc(1*sizeof(struct node));
rear->ptr = NULL;
rear->info = data;
```

```
front = rear;
}
else
{
temp=(struct node *)malloc(1*sizeof(struct node));
rear->ptr = temp;
temp->info = data;
temp->ptr = NULL;

rear = temp;
}

}


void display()
{
front1 = front;

if ((front1 == NULL) && (rear == NULL))
{
printf("Queue is empty");
return;
}
while (front1 != rear)
{
printf("%d ", front1->info);
front1 = front1->ptr;
}
if (front1 == rear)
printf("%d", front1->info);
}


void deq()
```

```
{
front1 = front;

if (front1 == NULL)
{
printf("\n queue is empty");
return;
}
else
if (front1->ptr != NULL)
{
front1 = front1->ptr;
printf("\n Dequed value : %d", front->info);
free(front);
front = front1;
}
else
{
printf("\n Dequed value : %d", front->info);
free(front);
front = NULL;
rear = NULL;
}

}

```

```
 1 - Enque
 2 - Deque
 3 - Display
 4 - Exit
 Enter choice : 1
Enter data : 22

 Enter choice : 1
Enter data : 33

 Enter choice : 1
Enter data : 55

 Enter choice : 2

 Dequed value : 22
 Enter choice : 3
33 55
 Enter choice : 4


...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM 9:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
struct node *prev;
};
struct node *head=NULL;
void insertleft()
{
struct node *new_node;
new_node=(struct node*)malloc(sizeof(struct node));
printf("Enter the number: \n");
scanf("%d",&new_node->data);
new_node->next=NULL;
new_node->prev=NULL;

if(head==NULL)
{
head=new_node;
}
else
{
new_node->next=head;
head->prev=new_node;
head=new_node;
}

}
```

```
void del()
{
struct node *temp;
int elem;
if(head==NULL)
{
printf("Empty List \n");
return;
}
printf("Enter the element to be deleted\n");
scanf("%d",&elem);
temp=head;
while(temp->data!=elem)
{
temp=temp->next;
if(temp==NULL)
{
printf("Element is not in the list\n");
return;
}
}
if(temp==head)
{
head=head->next;
}
else if(temp->next==NULL)
{
temp=temp->prev;
temp->next=NULL;
}

else
{
temp->prev->next=temp->next;
temp->next->prev=temp->prev;
```

```
}
}
void insert_betweenL()
{
int listele;
struct node *new_node,*temp;
printf("Enter the element in the list\n");
scanf("%d",&listele);
new_node=(struct node*)malloc(sizeof(struct
node));
printf("Enter the new node data\n");
scanf("%d",&new_node->data);
new_node->next=NULL;
new_node->prev=NULL;
if(head==NULL)
{
printf("Empty list\n"); return;
}
temp=head;
while(temp->data!=listele)
{
temp=temp->next;
if(temp==NULL)
{
printf("Element is  not in the list");
return;
}
}
new_node->prev=temp->prev;
temp->prev=new_node;
new_node->next=temp;
new_node->prev->next=new_node;
}

void insert_betweenR()
{
```

```
int listele;
struct node *new_node,*temp;
printf("Enter the element in the list\n");
scanf("%d",&listele);
new_node=(struct node*)malloc(sizeof(struct
node));
printf("Enter the new node data\n");
scanf("%d",&new_node->data);
new_node->next=NULL;
new_node->prev=NULL;
if(head==NULL)
{
printf("Empty list\n"); return;
}
temp=head;
while(temp->data!=listele)
{
temp=temp->next;
if(temp==NULL)
{
printf("Element is  not in the list");
return;
}
}
new_node->next=temp->next;
temp->next=new_node;
new_node->prev=temp;
new_node->next->prev=new_node;
}
void display()
{
struct node *temp;
temp=head;
while(temp!=NULL)
{
printf("%d\t",temp->data);
```

```
temp=temp->next;
}
printf("\n");
}


int main()
{
int choice;
while(1)
{

printf(" 1. Insert to the left in the beginning \n");
printf(" 2.insert a node before a given node\n");
printf(" 3.insert a node after a given node\n");
printf(" 4. delete \n");
printf(" 5. display\n");
printf(" 6. exit\n");
printf("\nEnter your choice: \n");
scanf("%d",&choice);
switch(choice)
{
case 1: insertleft(); break;
case 2:insert_betweenL();break;
case 3:insert_betweenR();break;
case 4: del(); break;
case 5: display(); break;
case 6: exit(0);
}
}
}
```

```
 1. Insert to the left in the beginning
 2.insert a node before a given node
 3.insert a node after a given node
 4. delete
 5. display
 6. exit

Enter your choice:
1
Enter the number:
44
 1. Insert to the left in the beginning
 2.insert a node before a given node
 3.insert a node after a given node
 4. delete
 5. display
 6. exit

Enter your choice:
5
44      33      22
```

```
 1. Insert to the left in the beginning
 2.insert a node before a given node
 3.insert a node after a given node
 4. delete
 5. display
 6. exit

Enter your choice:
2
Enter the element in the list
33
Enter the new node data
5
 1. Insert to the left in the beginning
 2.insert a node before a given node
 3.insert a node after a given node
 4. delete
 5. display
 6. exit

Enter your choice:
5
44      5       33      22
```

```
 1. Insert to the left in the beginning
 2.insert a node before a given node
 3.insert a node after a given node
 4. delete
 5. display
 6. exit

Enter your choice:
3
Enter the element in the list
33
Enter the new node data
6
 1. Insert to the left in the beginning
 2.insert a node before a given node
 3.insert a node after a given node
 4. delete
 5. display
 6. exit

Enter your choice:
5
44      5       33      6       22
 1. Insert to the left in the beginning
 2.insert a node before a given node
 3.insert a node after a given node
```

```
 1. Insert to the left in the beginning
 2.insert a node before a given node
 3.insert a node after a given node
 4. delete
 5. display
 6. exit

Enter your choice:
1
Enter the number:
22
 1. Insert to the left in the beginning
 2.insert a node before a given node
 3.insert a node after a given node
 4. delete
 5. display
 6. exit

Enter your choice:
1
Enter the number:
33
```

```
 1. Insert to the left in the beginning
 2.insert a node before a given node
 3.insert a node after a given node
 4. delete
 5. display
 6. exit

Enter your choice:
4
Enter the element to be deleted
22
 1. Insert to the left in the beginning
 2.insert a node before a given node
 3.insert a node after a given node
 4. delete
 5. display
 6. exit

Enter your choice:
5
44      5       33      6
```

```
 1. Insert to the left in the beginning
 2.insert a node before a given node
 3.insert a node after a given node
 4. delete
 5. display
 6. exit

Enter your choice:
6


...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM 10:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node* left;
struct node* right;
}*root1;

struct node *create()

{

struct node *temp;

printf("\n Enter data:");

temp=(struct node*)malloc(sizeof(struct node));

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

temp->left=temp->right=NULL;

return temp;

}

void insert(struct node *root,struct 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 Postorder(struct node* node)
{
if (node == NULL)
return;


Postorder(node->left);
```

```
Postorder(node->right);

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

void Inorder(struct node* node)
{
if (node == NULL)
return;

Inorder(node->left);

printf("%d ", node->data);

Inorder(node->right);
}

void Preorder(struct node* node)
{
if (node == NULL)
return;

printf("%d ", node->data);

Preorder(node->left);

Preorder(node->right);
```

```
}
int main()
{
int ch;
struct node *temp;
do
{
printf("1.create\n2.insert\n3.preorder\n4.postorder\n5.inorder\n6.Exit\n");
scanf("%d",&ch);
switch(ch)
{
case 1:
root1=create();
break;
case 2:
printf("enter the elem to be entered\n");
temp=(struct node*)malloc(sizeof(struct node));
scanf("%d",&temp->data);
insert(root1,temp);
break;
case 3:
Preorder(root1);
printf("\n");
break;
case 4:
Postorder(root1);
printf("\n");
break;
case 5:
Inorder(root1);
printf("\n");
break;
case 6:
break;
default:
```

```
                printf("wrong entry");
                }
                }while(ch!=6);
                
                
                }
```

```
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
1

 Enter data:12
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
2
enter the elem to be entered
13
1.create
2.insert
3.preorder
4.postorder
5.inorder
```

```
2
enter the elem to be entered
14
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
2
enter the elem to be entered
2
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
2
enter the elem to be entered
4
```

```
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
3
12 2 4 13 14
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
4
4 2 14 13 12
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
5
```

```
5.inorder
6.Exit
4
4 2 14 13 12
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
5
2 4 12 13 14
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
6
```