LAB PROGRAMS

1. Write a C program to implement stack using static array?

```
 #include<stdio.h>
#include<stdlib.h>
#define SIZE 6
int stack[SIZE];
int top = -1;



int isEmpty() {
return top == -1;
}
int isFull() {
return top == SIZE - 1;
}
void push(int ele) {
if (isFull()) {
printf("stack overflow %d", ele);
return;
}
stack[++top] = ele;
```

```
printf(“%d pushed to stack\n”, ele); }



int pop() {

if (isEmpty()) {

printf(“stack underflow\n”);

exit(EXIT_FAILURE);

}

int ele = stack[top--];

return ele;

}

int peek() {

if(isEmpty()) {

printf(“stack is empty\n”);

exit(EXIT_FAILURE);

}

return stack[top];

}

void display() {

if (isEmpty()) {

printf(“stack is empty”);
return;

}
```

```
printf(“stack elements ”);
for (int i = top; i >= 0;i--) {
printf(“%d”, stack[i]);
}
printf(“\n”);
}

int main() {
push(5);
push(10);
push(15);
display();

printf(“%d popped element\n”, pop);
display();
push(20);
push(30);
push(40);
printf(“Top element %d”, peek);
display();
return 0;
}
```

2.Write a C program to implement stack using dynamic array?

```
#include<stdio.h>

#include<stdlib.h>

int *stack,size,top=-1;

int isempty();

int isfull();

void push(int);

int pop();

void display();

void topelement();

int main()

{

int ch,x;

printf("enter initial size of the stack: ");

scanf("%d",&size);

stack=(int*)calloc(sizeof(int),size);

do

 {
printf("1.PUSH 2.POP 3.TOP ELEMENT 4.DISPLAY 5.EXIT

\n"); printf("PLEASE ENTER YOUR CHOICE : ");

scanf("%d",&ch);
switch(ch)

{

case 1 :printf("ENTER THE ELEMENT :\n");

          scanf("%d",&x);

          push(x);

          break;

case 2 :x=pop();

          if(x==0)

               printf("STACK UNDERFLOW !\n");


          else

               printf("POPPED ELEMENT IS %d \n",x);


          break;

case 3 : topelement();

          break;

case 4 : display();

          break;

default : printf("INVALID INPUT !\n");
}

}while(ch>0 &&

ch<5); free(stack);

return 0;
```

```
}

int isfull()

{

   if(top == size-1)

        return 1;

   else

        return 0;

}

int isempty()

{

if(top == -1)

   return 1;

else

   return 0;

}

void push(int e)

{

   if(isfull())
   {

      stack=(int*)realloc(stack,2*size);

      printf("size of the stack is

   increased\n"); }

      top++;
```

```
        stack[top]=e;

        printf("%d is pushed into the

stack\n",e); }

int pop()

{

if(isempty())

{

    return 0;

}

int x=stack[top];

top--;

return x;

}

void topelement()

{

if(isempty())

    printf("STACK IS EMPTY !");
else

    printf("Top Element element is %d \n",stack[top]);

}

void display()

{

if(isempty())
```

```
{
   printf("STACK IS EMPTY ! \n");
}
printf("STACK ELEMENTS ARE : \n");
for(int i=top;i>=0;i--)
   printf("%d\n",stack[i]);
}
```

3. Write a C program to convert infix to postfix using stack?

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

#define MAX 30

char stack[MAX];

int top;
void push(char);

char pop();

void

infixtopostfix(char*); int

priority(char ch); void

push(char x)

 { if (top != MAX-1)
```

```
  {
   top++;
   stack[top] = x;
} }
char pop()
 { char x;
if(top!=-1)
{
   x = stack[top];
   top--;
   return x;
} }
int isempty()
 { if (top==-1)
   return 1;
   return 0;
}
int priority(char ch);
// infix to postfix conversion
void infixtopostfix(char *a)
 { int x,j=0;
char b[20]; //for storing resultant postfix
expression int len=strlen(a);
```

```
for(int i=0;i<len;i++)
 { char ch=a[i];
   if(isalnum(ch)) //if a[i] is operand
      b[j++]=ch;
   else if(ch=='(')
      push(ch);
   else if(ch==')')
    { while(stack[top]!='(')
         b[j++]=pop();
      pop(); //popping '('
    }
   else //if a[i] is operator
    {
   if(ch=='('|| isempty()) //if stack top has '(' or stack is
      empty push(ch);
   else
    {
      while(priority(ch)<=priority(stack[top]))
         b[j++]=pop();
      push(ch);
    } }
}//end of for
```

```
while(isempty()==0)

   b[j++]=pop();

b[j]='\0';

printf("postfix expression is.:

%s",b); }

int priority(char ch)

{

switch(ch)

{

case '$':return(3);

case '*':

case '/':

case '%':return(2);

case '+':

case '-':return(1);
} }

int main()

{

char p[20];

printf("enter proper infix expression : ");

scanf("%s",p);

infixtopostfix(p);

return 0;
```

```
}
```

4. Write a C program to implement evaluation of postfix expression?

```
#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<math.h>

#define MAX 30

int stack[MAX];

int top=-1;

char pop();

void evaluate(char *);

void push(char x)

 { if(top!=MAX-1)
    {

       top++;

       stack[top]=x;

    } }

char pop()

 { char x;

   if(top!=-1)

    { x=stack[top];
```

```
       top--;

       return x;

    } }

int isempty()

 { if(top==-1)

       return 1;

    else

       return 0;

}

void evaluate(char

 *a) { int i;

    char ch;

    int op1,op2;

    for(i=0;a[i]!='\0';i++)
     { ch=a[i];

       if(isdigit(ch))

          push(ch-48);

       else

        { switch(ch){

              case '$':op2=pop();

                  op1=pop();

                  push(pow(op1,op2));

                  break;
```

```
           case '*':op2=pop();

               op1=pop();

               push(op1*op2);

               break;

          case '/':op2=pop();

               op1=pop();

               push(op1/op2);

               break;

          case '%':op2=pop();

              op1=pop();

              push(op1%op2);

                break;

          case '+':op2=pop();
                op1=pop();

               push(op1+op2);

               break;

          case'-':op2=pop();

                      op1=pop();

              push(op1-op2);

              break;

          default:printf("wrong input\n");

      } } }

printf("result=%d\n",pop());
```

```
}
void main()
 { char p[20];
   printf("enter a postfix expression with digits as
   operands\n"); scanf("%s",p);
   evaluate(p);
}
```

5.Write a C program to implement queue using arrays?

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
int queue[SIZE];
int front = -1, rear = -1;
void enqueue(int ele)
{
      if (rear == SIZE - 1)
 {
            printf("Queue is Full \n");
      }
else
{
            if (front == -1)
```

```
                        front = 0;

                rear++;

                queue[rear] = ele;

                printf("%d inserted ele into the queue.\n",

        ele); }

}

void dequeue()

{

        if (front == -1 || front > rear) {

                printf("Queue is Empty \n");
}

 else

{

                printf("%d deleted ele from the queue.\n",

                queue[front]); front++;

        }

}

void display() {

        if (front == -1 || front > rear) {

                printf("Queue is Empty\n");

        }

 else

{
```

```
            printf("\nQueue elements are: ");
             for (int i = front; i <= rear; i++)
{
            printf("%d ", queue[i]);
             }
             printf("\n");
        }
}
int main()
{
        int ch, ele;

        while (1) {
             printf("\n----- Queue Operations -----\n");
             printf("1. Enqueue\n2. Dequeue\n3. Display\n4.
             Exit\n"); printf("Enter your choice: ");
             scanf("%d", &ch);

             switch (ch) {
                    case 1:
                          printf("Enter value to insert: ");
                          scanf("%d", &ele);
                          enqueue(ele);
```

```
                        break;

                case 2:
                        dequeue();
                        break;

                case 3:
                        display();
                        break;

                case 4:
                        printf("Exiting program.\n");
                        exit(0);

                default:
                       printf("Invalid choice! Please try again.\n");
            }
        }
        return 0;
}
```

6. Write a C program to implement circular queue using arrays?

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

```
#include <stdlib.h>

#define SIZE 5


int CQ[SIZE];

int front = -1, rear = -1;
void enqueue(int ele)

{

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

1)) {

                printf("Queue is Full \n");

        }

else

{

                if (front == -1)

                        front = 0;

                rear = (rear + 1) % SIZE;

                CQ[rear] = ele;

                printf("%d inserted ele into the queue.\n", ele); }

}

void dequeue()

 {

        if (front == -1)

{
```

```
                printf("Queue is Empty \n");
        }
 else
{
                printf("%d deleted ele from the queue.\n",
                CQ[front]); if (front == rear) {
                      front = rear = -1;
                }
 else
 {
              front = (front + 1) % SIZE;
                }
        }
}
void peek()
{
        if (front == -1)
{
                printf("Queue is Empty!\n");
        }
else
{
                printf("Front element is: %d\n", CQ[front]);
        }
}

void display() {

        if (front == -1)

 {

                printf("Queue is Empty!\n");

        }

else

{

                printf("\nQueue elements are:

                "); int i = front;

                while (1) {

                        printf("%d ", CQ[i]);

                        if (i == rear)

                                break;

                        i = (i + 1) % SIZE;

                }

                printf("\n");

        }

}



void isEmpty()
```

```
{
        if (front == -1)
                printf("Queue is Empty.\n");
        else
                printf("Queue is NOT Empty.\n");
}
void isFull() {
        if ((front == 0 && rear == SIZE - 1) || (front == rear +
                1)) printf("Queue is Full.\n");
        else
                printf("\nQueue is NOT Full.\n");
}


int main()
{
        int ch, ele;


        while (1) {
                printf("\n***Circular Queue Operations
                ***\n"); printf("1. Enqueue (Insert)\n");
                printf("2. Dequeue (Delete)\n");
                printf("3. Peek (Front Element)\n");
                printf("4. Display Queue\n");
                printf("5. Check if Queue is Empty\n");

                printf("6. Check if Queue is Full\n");

                printf("7. Exit\n");

                printf("***********************\n");

                printf("Enter your choice: ");

                scanf("%d", &ch);


                switch (ch)

            {

                    case 1:

                        printf("Enter element to insert: ");

                        scanf("%d", &ele);

                        enqueue(ele);

                        break;


                    case 2:

                        dequeue();

                        break;


                    case 3:

                        peek();
                        break;
                    case 4:
                        display();
                        break;

                    case 5:
                        isEmpty();
                        break;

                    case 6:
                        isFull();
                        break;

                    case 7:
                        printf("Exiting program.\n");
                        exit(0);

                    default:
                       printf("Invalid choice! Please try again.\n");
                }
            }
            return 0;
            }
```

7.Write a C program to implement stack using linked list?

```
#include <stdio.h>

#include <stdlib.h>



struct node {

    int data;

    struct node* next;

};


struct node* top = NULL;



void push(int ele) {

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

    if (!newNode) {

        printf("Stack Overflow (Memory not allocated)\n");

        return;

    }

    newnode->data = ele;

    newnode->next = top;

    top = newnode;

    printf("%d inserted at beg\n", ele);
}



int pop() {
    if (top == NULL) {
        printf("Stack Underflow\n");
        return -1;
    }
    struct node* temp = top;
      top = top->next;
    free(temp);


}

int peek() {
    if (top == NULL) {
        printf("Stack is empty\n");
        return -1;
    }
    return top->data;
}
```

```
void display() {
    if (top == NULL) {
        printf("Stack is empty\n");
        return;
    }
    struct node* temp = top;
    printf("elements in stack are: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
int main() {
    int ch, ele;

    while (1) {
        printf("\n~~~~ Stack Menu ~~~~\n");
        printf("1. Push\n2.Pop\n3.Peek\n4.Display\n5.Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &ch);
```

```
    switch (ch)
 {
        case 1:
            printf("Enter element to push: ");
            scanf("%d", &ele);
            push(ele);
            break;

        case 2:
             pop()=ele;
            if (ele!= -1)
                printf("deleted  %d\n", ele);
            break;

        case 3:
            peek()=ele;
            if (ele != -1)
                printf("Top element: %d\n", ele);
            break;

        case 4:
            display();
```

```
                break;

            case 5:
                printf("Exiting...\n");
                exit(0);

            default:
                printf("Invalid choice! Try again.\n");
        }
    }

    return 0;
}
```

8.Write a C program to implement Queue using linked list?

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

struct node {
    int data;
    struct node* next;
};
```

```
struct node* front = NULL;

struct node* rear = NULL;



void enqueue(int ele) {

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

   newnode->data = ele;

   newnode->next = NULL;


   if (front == NULL) {


      front = rear = newnode;

   }

 else

 {

      rear->next = newnode;

      rear = newnode;

   }

   printf("%d inserted  to queue.\n", ele);

}
```

```
void dequeue() {
    if (front == NULL) {
        printf("Queue is empty\n");
        return;
    }
    struct node* temp = front;
    printf("%d deleted  from queue\n", front->data);

    front = front->next;
    free(temp);

    if (front == NULL) {
        rear = NULL;
    }
}
void peek() {
    if (front == NULL) {
        printf("Queue is empty.\n");
        return;
    }
    printf("Front element: %d\n", front->data);
```

```
}
void display() {
    if (front == NULL)
{
        printf("Queue is empty.\n");
        return;
    }

    struct node* temp = front;
    printf("Queue: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    int ch, ele;

    while (1) {
        printf("\n***Queue Menu ***\n");
        printf("1. Enqueue\n2.Dequeue\n3.Peek\n4.Display\n5.Exit\n");
       printf("Enter your choice: ");
        scanf("%d", &ch);
    switch (ch)
  {
        case 1:
            printf("Enter value to insert: ");
            scanf("%d", &ele);
            enqueue(ele);
            break;
        case 2:
            dequeue();
            break;
        case 3:
            peek();
            break;
        case 4:
            display();
            break;
        case 5:
            printf("Exiting...\n");
            exit(0);
        default:
            printf("Invalid choice! Please try again.\n");
        }
    }
    return 0;
}
```

9.Write a C program to implement double linked list?

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

struct node {
    int data;
    struct node* prev;
    struct node* next;
};

struct node* head = NULL;
void insertBeg(int ele) {
    struct node* newnode = (struct node*)malloc(sizeof(struct node));
    newnode->data=ele;
    if (head == NULL)
    {
        head = newnode;
        return;
    }
    newnode->next = head;
    head->prev = newnode;
    head = newnode;
}


void insertEnd(int ele) {
    struct node* newnode =(struct node*)malloc(sizeof(struct node));
    newnode->data=ele;
    if (head == NULL) {
        head = newnode;
        return;
    }
    struct node* temp = head;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newnode;
    newnode->prev = temp;
```

```
}


void insertPos(int ele, int pos) {
    if (pos == 1) {
        insertBeg(ele);
        return;
    }

    struct node* temp = head;
    for (int i = 1; i < pos - 1 && temp != NULL; i++)
        temp = temp->next;

    if (temp == NULL) {
        printf("Position out of bounds!\n");
        return;
    }

    struct node* newnode = (struct node*)malloc(sizeof(struct node));
    newnode->data=ele;
    newnode->next = temp->next;
    newnode->prev = temp;
    if (temp->next != NULL)
        temp->next->prev = newnode;

    temp->next = newnode;
}



void deleteBeg() {
    if (head == NULL) {
        printf("List is empty!\n");
        return;
    }
    struct node* temp = head;
    head = head->next;

    if (head != NULL)
        head->prev = NULL;

    free(temp);
}

void deleteEnd() {
    if (head == NULL) {
        printf("List is empty!\n");
        return;
    }
    struct node* temp = head;

    if (temp->next == NULL) {
        head = NULL;
        free(temp);
        return;
    }

    while (temp->next != NULL)
        temp = temp->next;

    temp->prev->next = NULL;
    free(temp);
}
void deletePos(int pos) {
    if (head == NULL) {
        printf("List is empty!\n");
        return;
    }

    struct node* temp = head;

    if (pos == 1) {
        deleteBeg();
        return;
    }

    for (int i = 1; i < pos && temp != NULL; i++)
        temp = temp->next;

    if (temp == NULL) {
        printf("Position not found!\n");
        return;
    }

    if (temp->next != NULL)
        temp->next->prev = temp->prev;

    if (temp->prev != NULL)
        temp->prev->next = temp->next;
    free(temp);
}


void displayForward() {
    struct node* temp = head;
    printf("List: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}


void displayBackward()
{
    if (head == NULL)
    {
        printf("List is empty!\n");
        return;
    }
    struct node* temp = head;
    while (temp->next != NULL)
        temp = temp->next;

    printf("List: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->prev;
    }
    printf("\n");
}

int main() {
    int ch, ele, pos;

    while (1) {
        printf("\n*** Doubly Linked List Menu ***\n");
        printf("1. Insert at Beginning\n2.Insertion at end\n3.Insertion at
specific position\n4.Deletion at the beginning\n5.Deletion at
end\n6.Deletion at position\n7.Display forward\n8.Display
backward\n9.Exit\n");

       printf("Enter choice: ");
```

```
scanf("%d", &ch);

switch (ch) {
    case 1:
        printf("Enter value: ");
        scanf("%d", &ele);
        insertBeg(ele);
        break;

    case 2:
        printf("Enter value: ");
        scanf("%d", &ele);
        insertEnd(ele);
        break;

    case 3:
        printf("Enter value and position: ");
        scanf("%d %d", &ele, &pos);
        insertPos(ele, pos);
        break;

    case 4:
```

```
    deleteBeg();
    break;

case 5:
    deleteEnd();
    break;

case 6:
    printf("Enter position: ");
    scanf("%d", &pos);
    deletePos(pos);
    break;

case 7:
    displayForward();
    break;

case 8:
    displayBackward();
    break;

case 9:
```

```
                exit(0);


            default:
                printf("Invalid choice!\n");
        }
    }

    return 0;
}
```