25. Write a C Program using structure for entering details of the five students as name, admission number, Date of birth, department and display all the details.

```
#include<stdio.h>
int main()
{
struct student
{
int roll_num;
char name[60];
int fee;
char DOB[80];
};
struct student stu[90];
int a,b;
printf("\n ENTER THE NUMBER OF STUDENTS");
scanf("%d",&a);
for(b=0;b<a;b++)
{
printf("\n ENTER THE ROLL NUMBER");
scanf("%d",&stu[b].roll_num);
printf("\n ENTER THE NAME");
scanf("%s",&stu[b].name);
printf("\n ENTER THE FEE");
scanf("%d",&stu[b].fee);
printf("\n ENTER THE DOB");
scanf("%s",&stu[b].DOB);
}
for(b=0;b<a;b++)
{
printf("\n DETAILS OF THE STUDENT YOU HAVE ASKED FOR %d",b+1);
printf("\n ROLL NO = %d",stu[b].roll_num);
printf("\n NAME = %s",stu[b].name);
printf("\n FEE = %d",stu[b].fee);
printf("\n DOB = %s",stu[b].DOB);
}
}
```

OUTPUT

```
 DETAILS OF THE STUDENT YOU HAVE ASKED FOR 1
 ROLL NO = 20
 NAME = GEETIKA
 FEE = 9000
 DOB = 21/11/2001
 DETAILS OF THE STUDENT YOU HAVE ASKED FOR 2
```

```
ROLL NO = 30
NAME = HARSHITHA
FEE = 6000
DOB = 24/02/2002
DETAILS OF THE STUDENT YOU HAVE ASKED FOR 3
ROLL NO = 14
NAME = KUSUMA
FEE = 8000
DOB = 23/11/2002
DETAILS OF THE STUDENT YOU HAVE ASKED FOR 4
ROLL NO = 50
NAME = VAISHNAVI
FEE = 8500
DOB = 21/03/2002
DETAILS OF THE STUDENT YOU HAVE ASKED FOR 5
ROLL NO = 42
NAME = SAKITHA
FEE = 6200
DOB = 12/06/2002
```

26. Write a C program to find length of string using pointers.

```
#include<stdio.h>
int stringlength(char*);
void main()
{
char string[20];
int length;
printf("\n THE STRING TO BE ENTERED : ");
gets(string);
length = stringlength(string);
printf("THE STRING LENGTH TO BE ENTERED %s IS : %d", string, length);
}
int stringlength(char*p)
{
int count = 0;
while (*p != '\0') {
count++;
p++;
}
return count;
}
```

OUTPUT

```
THE STRING TO BE ENTERED: GEETIKA
```

THE STRING LENGTH TO BE ENTERED GEETIKA IS : 6

27.Write a C program to copy one string to another using pointers.

```
 #include<stdio.h>
#define MAX 100
int main()
{
char str[MAX],copy_str[MAX];
char*pstr,*pcopy_str;
int i=0;
pstr=str;
pcopy_str=copy_str;
printf("\n THE STRING TO BE ENTERED");
gets(str);
while(*pstr!='\0')
{
*pcopy_str=*pstr;
 pstr++,pcopy_str++;
}
*pcopy_str='\0';
printf("\n THE COPIED STRING IS");
pcopy_str= copy_str;
while(*pcopy_str!='\0')
{
  printf("%c",*pcopy_str);
pcopy_str++;
}
}
```

OUTPUT

```
 THE STRING TO BE ENTERED VIJAYAWADA
 THE COPIED STRING IS VIJAYAWADA
```

28. Write a C program to compare two strings using pointers.

```
#include<stdio.h>
#define MAX 100
int main()
{
char string1[MAX],string2[MAX],*str1,*str2;
int i,equal = 0;
printf("THE STRING TO BE ENTERED: ");
scanf("%s",string1);
printf("THE 2ND STRING TO BE ENTERED: ");
scanf("%s",string2);
```

```
str1 = string1;
str2 = string2;
while(*str1 == *str2)
{

if ( *str1 == '\0' || *str2 == '\0' )
break;
str1++;
str2++;
}
if( *str1 == '\0' && *str2 == '\0' )
printf("\n\nTHE STRINGS ARE SAME.");
else
printf("\n\nTHE STRINGS ARE NOT SAME.");
}
```

OUTPUT

```
 THE STRING TO BE ENTERED:MEDICINE
THE 2ND STRING TO BE ENTERED:MECHANICAL
THE STRINGS ARE NOT SAME
```

29. Write a C program to find the reverse of a string recursively and non recursively.

A)

```
#include <stdio.h>
#include <string.h>
void reversestring(char*, int, int);

int main()
{
char stringarray[120];
printf("THE STRING TO BE ENTERED:");
scanf("%s", &stringarray);

reversestring(stringarray, 0, strlen(stringarray)-1);
printf("\nTHE STRING REVERSED IS: %s",stringarray);
return 0;
}

void reversestring(char *x, int begin, int cease)
{
char ch;
if (begin >= cease)
return;
ch = *(x+begin);
    *(x+begin) = *(x+cease);
```

```
    *(x+cease) = ch;
reversestring(x, ++begin, --cease);
}
```
OUTPUT
```
 THE STRING TO BE ENTERED:KARUMUDI
THE STRING REVERSED IS:IDMURAK
```
B)
```
#include <stdio.h>
#include <string.h>
#define MAX 100
int main()
{
char string[MAX],temp;
int i=0,j=0;
printf("\nTHE STRING TO BE ENTERED");
gets(string);
j=strlen(string)-1;
while(i<j)
{
temp = string[j];
string[j]=string[i];
string[i]=temp;
i++;
j--;
}
printf("\n THE STRING REVERSED IS ");
puts(string);
}
```
OUTPUT
```
THE STRING TO BE ENTERED COMPUTER SCIENCE
 THE STRING REVERSED IS ECNEICS RETUPMOC
```

30. Create a binary tree and output the data with 3 tree traversals.
```
 #include <stdio.h>
#include <stdlib.h>
struct node
{
int data;
struct node* left;
struct node* right;
};
struct node* newNode(int data)
{
struct node* node = (struct node*)
```

```
malloc(sizeof(struct node));
node->data = data;
node->left = NULL;
node->right = NULL;
return(node);
}
void printPostorder(struct node* node)
{
if (node == NULL)
return;
printPostorder(node->left);
printPostorder(node->right);
printf("%d ", node->data);
}
void printInorder(struct node* node)
{
if (node == NULL)
return;
printInorder(node->left);
printf("%d ", node->data);
printInorder(node->right);
}
void printPreorder(struct node* node)
{
if (node == NULL)
return;
printf("%d ", node->data);
printPreorder(node->left);
printPreorder(node->right);
}
int main()
{
struct node *root  = newNode(56);
root->left        = newNode(140);
root->right       = newNode(166);
root->left->left     = newNode(50);
root->left->right   = newNode(143);
printf("\nBINARY TREE IN PRE-ORDER TRAVERSAL IS \n");
printPreorder(root);
  printf("\nBINARY TREE IN INORDER TRAVERSAL IS \n");
printInorder(root);

printf("\nBINARY TREE IN POST-ORDER TRAVERSAL IS \n");
```

```
printPostorder(root);

getchar();
return 0;
}
```

OUTPUT

```
 BINARY TREE IN PRE-ORDER TRAVERSAL IS
 60 32 160 89 144
BINARY TREE IN INORDER TRAVERSAL IS
 160 32 89 60 144
BINARY TREE IN POST-ORDER TRAVERSAL IS
 160 89 32 144 60
```

34. Write a C program to implement the STACK operation using array as a data structure. Users must be given the following choices to perform relevant tasks.

a. Push an element on to the STACK.
b. Pop and element from the STACK.
c. Peek the STACK.
d. Display the STACK.
e. Exit the program.

```
 #include<stdio.h>
#define MAX 50
int stack[MAX],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
void peek(void);
int main()
{
top=-1;
printf("\n THE SIZE OF THE STACK TO BE ENTERED:");
scanf("%d",&n);
printf("\n\t STACK OPERATIONS APPLIED IN THIS ARRAY");
printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.PEEK\n\t 5.EXIT");
do
{
printf("\n THE CHOICE TO BE ENTERED:");
scanf("%d",&choice);
switch(choice)
{
case 1:
{
```

```
push();
break;
}
case 2:
{
pop();
break;
}
case 3:
{
display();
break;
}
case 4:
{
peek();
break;
}
case 5:
{
printf("\n\t EXIT POINT ");
break;
}
default:
{
printf ("\n\t YOU HAVE ENTERED A WRONG ELEMENT");
}
}
}
while(choice!=5);
return 0;
}
void push()
{
if(top>=n-1)
{
printf("\n\tSTACK OVER-FLOW");
}
else
{
printf(" TYPE AN ELEMENT THAT NEEDED TO BE PUSHED:");
scanf("%d",&x);
top++;
```

```
stack[top]=x;
}
}
void pop()
{
if(top<=-1)
{
printf("\n\t STACK UNDER-FLOW");
}
else
{
printf("\n\t THE ELEMENT WHICH IS POPPED %d",stack[top]);
top--;
}
}
void display()
{
if(top>=0)
{
printf("\n THE ELEMENTS IN STACK \n");
for(i=top; i>=0; i--)
printf("\n%d",stack[i]);
printf("\n ENTER NEXT CHOICE");
}
else
{
printf("\n The STACK IS EMPTY");
}

}
void peek()
{
printf("\nPEEK ELEMENT IS %d",stack[top]);
}
```

OUTPUT

```
    THE SIZE OF THE STACK TO BE ENTERED:6
    STACK OPERATIONS APPLIED IN THIS ARRAY
       1.PUSH
       2.POP
       3.DISPLAY
       4.PEEK
       5.EXIT
 THE CHOICE TO BE ENTERED:2
```

```
 TYPE AN ELEMENT THAT NEEDED TO BE PUSHED:32
  THE CHOICE TO BE ENTERED:6
   THE ELEMENTS IN STACK
    32
   ENTER NEXT CHOICE
 THE CHOICE TO BE ENTERED:2
   EXIT POINT
```

35. Write a C program to reverse a string using STACK.

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

#define max 100
int top,stack[max];

void push(char x)
{
if(top == max-1)
{
printf("stack overflow");
}
else
{
stack[++top]=x;
}
}
void pop()
{
printf("%c",stack[top--]);
}
main()
{
char str[60];
printf("The string to be entered : \n");
scanf("%s",&str);
int len = strlen(str);
int i;
for(i=0;i<len;i++)
push(str[i]);
for(i=0;i<len;i++)
pop();
}
```

OUTPUT

The string to be entered :
honey
yenoh

36. Write a C program to convert the given infix expression to post-fix expression using STACK.

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#define MAX 70

char st[MAX];
int top =-1;
void push(char st[],char);
char pop(char st[]);
void InfixtoPostfix(char source[],char target[]);
int getPriority(char);
int main()
{
char infix[100], postfix[100];
   printf("\n the expression of infix to be entered");
   gets(infix);
   strcpy(postfix,"");
InfixtoPostfix(infix,postfix);
   printf("\n expression of postfix is ");
   puts(postfix);
return 0;
}
void InfixtoPostfix(char source[],char target[])
{
int i=0,j=0;
char temp;
   strcpy(target,"");
while(source[i]!='\0')
{
if(source[i]=='(')
{
push(st,source[i]);
i++;
}
else if(source[i]==')')
{
while((top!=-1)&&(st[top]!='('))
```

```
{
target[j]=pop(st);
j++;
}
if(top==-1)
{
   printf("\n incorrect expression");
   exit(1);
}
temp= pop(st);
i++;
}
else if(isdigit(source[i])|| isalpha(source[i]))
{
target[j]=source[i];
j++;
i++;
}
else if(source[i]=='+'|| source[i]=='-'|| source[i]=='*'|| source[i]== '/'|| source[i]=='%')
{
while((top!=-1)&&(st[top]!='(')&&
     (getPriority(st[top])>getPriority
     (source[i])))
{
target[j]=pop(st);
j++;
}
push(st,source[i]);
i++;
}
else
{
   printf("\n incorrect elemnet is present in the expression");
   exit(1);
}
}
while((top!=-1)&&(st[top]!='('))
{
target[j]=pop(st);
j++;
}
target[j]='\0';
}
```

```
int getPriority (char op)
{
if(op=='/'||op=='*'||op=='%')
return 1;
else if(op=='+'||op=='-')
return 0;
}
void push(char st[],char val)
{
if(top==MAX -1)
   printf("\n overflow of stack");
else
{
top++;
st[top]=val;
}
}
char pop(char st[])
{
char val=' ';
if(top==-1)
   printf("\n underflow of stack");
else
{
val=st[top];
top--;
}
return val;
}
```

OUTPUT

```
 The expression of infix to be entered (U-V)*(X-Y)
   expression of postfix is UV-XY-*
```

38. Write a C program to evaluate the given pre-fix expression and post-fix Expressions.

```
 #include<stdio.h>

int stack[20];
int top = -1;


void push(int x)
{
```

```
stack[++top] = x;
}

int pop()
{
return stack[top--];
}

int main()
{
char exp[20];
char *e;
int n1,n2,n3,num;
printf("ENTER THE EXPRESSION : ");
scanf("%s",exp);
e = exp;
while(*e != '\0')
{
if(isdigit(*e))
{
num = *e - 48;
push(num);
}
else
{
n1 = pop();
n2 = pop();
switch(*e)
{
case '+':
{
n3 = n1 + n2;
break;

}
case '-':
{
n3 = n2 - n1;
break;

}
case '*':
{
```

```
n3 = n1 * n2;
break;

}
case '/':
{
n3 = n2 / n1;
break;

}
}
push(n3);
}
e++;
}
printf("\nTHE RESULT OF EXPRESSION %s  =  %d\n\n",exp,pop());
return 0;
}
```

OUTPUT

```
 THE EXPRESSION TO BE ENTERED : 642*6+3
 THE RESULT OF EXPRESSION 642*6+3=3
```

37. Write a C program to convert the given in-fix expression to pre-fix expression using STACK.

```
 #include<stdio.h>
#include<string.h>
#include<ctype.h>
#define MAX 50

char st[MAX];
int top=-1;
void reverse(char str[]);
void push(char st[],char);
char pop(char st[]);
void Infixtopostfix(char source[],char target[]);
int getPriority(char);
char infix[100],postfix[100],temp[100];
int main()
{
printf("\n ENTER THE INFIX EXPRESSION");
gets(infix);
reverse(infix);
strcpy(postfix,"");
```

```
Infixtopostfix(temp,postfix);
printf("\n THE CORRESPONDING POSTFIX EXPRESSION");
puts(postfix);
strcpy(temp,"");
reverse(postfix);
printf("\n THE PREFIX EXPRESSION");
puts(temp);
return 0;
}
void reverse(char str[])
{
int len,i=0,j=0;;
len=strlen(str);
j=len-1;
while(j>=0)
{
if(str[j]=='(')
  temp[i]=')';
else if(str[j]==')')
  temp[i]='(';
else
temp[i]=str[j];
i++;
j--;
}
temp[i]='\0';
}
void Infixtopostfix(char source[], char target[])
{
int i=0,j=0;
char temp;
strcpy(target,"");
while(source[i]!='\0')
{
if(source[i]=='(')
{
push(st, source[i]);
i++;
}
else if(source[i]==')')
{
while((top!=-1)&&(st[top]!='('))
{
```

```
target[j]=pop(st);
j++;
}
if(top==-1)
{
printf("\n INCORRECT EXPRESSION");
exit(1);
}
temp=pop(st);
i++;
}
else if(isdigit(source[i])||isalpha(source[i]))
{
target[j]= source[i];
j++;
i++;
}
else if(source[i]=='+'||source[i]=='-'
        ||source[i]=='*'||source[i]
        =='/'||source[i]=='%')
{
while((top!=-1)&&(st[top]!='(')
    &&(getPriority(st[top])>
    getPriority(source[i])))
{
target[j]= pop(st);
j++;
}
push(st, source[i]);
i++;
}
else
{
printf("\n INCORRECT ELEMENTS IN EXPRESSION");
exit(1);
}
}
while((top!=-1)&&(st[top]!='('))
{
target[j]= pop(st);
j++;
}
target[j]='\0';
```

```
}
int getPriority(char op)
{
if(op=='/'||op=='*'||op=='%')
return 1;
else if(op=='+'||op=='-')
return 0;
}
void push(char st[], char val)
{
if(top==MAX -1)
printf("\n STACK OVER-FLOW");
else
{
top++;
st[top]=val;
}
}
char pop(char st[])
{
char val= ' ';
if(top==-1)
printf("\n STACK UNDER-FLOW");
else
{
val=st[top];
top--;
}
return val;
}
```

OUTPUT

```
 ENTER THE INFIX EXPRESSION  U+V-X*Y
THE CORRESPONDING POSTFIX EXPRESSION   X*VU+-
THE PREFIX EXPRESSION -+UV*XY
```

39. Write a C program to implement a Linear-Queue, user must choose the following options:

a. Add an element to the Queue – EnQueue.

b. Remove an element from the Queue – DeQueue.

c. Display the elements of the Queue.

d. Terminate the program.

```
#include<stdio.h>
#define MAX 50
```

```
int queue[MAX];
int front=-1,rear=-1;
void insert(void);
int delete_element(void);
int peep(void);
void display(void);
int main()
{
int option,val;
do
{
printf("\n\n*****MAIN MENU*****");
printf("\n 1. ELEMENT TO BE ADDED");
printf("\n 2. ELEMENT TO BE DELETED");
printf("\n 3. PEEK");
printf("\n 4. QUEUE TO BE DISPLAYED");
printf("\n 5. EXIT");
printf("\n ********************");
printf("\n\n ENTER THE OPTION");
scanf("%d",&option);
switch(option)
{
case 1:
insert();
break;

case 2:
val=delete_element();
if(val!=-1)
printf("\n THE NUMBER TO BE DELETED IS %d",val);
break;

case 3:
val= peep();
if(val!=-1)
printf("\n THE FIRST ELEMENT IN THE QUEUE IS %d",val);
break;

case 4:
display();
break;
}
}while(option!=5);
```

```
return 0;
}
void insert()
{
int num;
printf("\n ENTER THE ELEMENT TO BE ADDED ");
scanf("%d",&num);
if(rear==MAX-1)
printf("\n OVER-FLOW");
else if(front==-1&&rear==-1)
front=rear=0;
else
rear++;
queue[rear]=num;
}
int delete_element()
{
int val;
if(front==-1 || front>rear)
{
printf("\n UNDER-FLOW");
return -1;
}
else
{
val=queue[front];
front++;
if(front>rear)
front=rear=-1;
return val;
}
}
int peep()
{
if(front==-1 || front> rear)
{
printf("\n QUEUE IS EMPTY");
return -1;
}
else
{
return queue[front];
}
```

```
}
void display()
{
int i;
printf("\n");
if(front==-1 || front > rear)
printf("\n QUEUE IS EMPTY");
else
{
for(i=front;i<=rear;i++)
printf("\t %d",queue[i]);
}
}
```

OUTPUT

```
 *****MAIN MENU*****
 1. ELEMENT TO BE ADDED
 2. ELEMENT TO BE DELETED
 3. PEEK
 4. QUEUE TO BE DISPLAYED
 5. EXIT
 ********************
 ENTER THE OPTION1
 ENTER THE ELEMENT TO BE ADDED 66
 *****MAIN MENU*****
 1. ELEMENT TO BE ADDED
 2. ELEMENT TO BE DELETED
 3. PEEK
 4. QUEUE TO BE DISPLAYED
 5. EXIT
 ********************
  ENTER THE OPTION3
THE FIRST ELEMENT IN THE QUEUE IS 66
 *****MAIN MENU*****
 1. ELEMENT TO BE ADDED
 2. ELEMENT TO BE DELETED
 3. PEEK
 4. QUEUE TO BE DISPLAYED
 5. EXIT
 ********************
  ENTER THE OPTION4
                66
 *****MAIN MENU*****
 1. ELEMENT TO BE ADDED
```

2. ELEMENT TO BE DELETED
3. PEEK
4. QUEUE TO BE DISPLAYED
5. EXIT
**********
ENTER THE OPTION5

40. Write a C program to implement a Circular-Queue, user must choose the following options:
a. Add an element to the Queue – EnQueue.
b. Remove an element from the Queue – DeQueue.
c. Display the elements of the Queue.
d. Terminate the program.

```
#include<stdio.h>
#define MAX 50

void insertq(int[], int);
void deleteq(int[]);
void display(int[]);

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

int main()
{
int n, ch;
int queue[MAX];
do
{
printf("\n\n CIRCULAR QUEUE CHOICES:\n1. ENQUEUE \n2. DEQUEUE\n3. DISPLAY\n0.
EXIT");
printf("\nPRESS THE CHOICE: ");
scanf("%d", &ch);
switch (ch)
{
case 1:
printf("\n THE NUMBER TO BE ENTERED: ");
scanf("%d", &n);
insertq(queue, n);
break;

case 2:
deleteq(queue);
break;
```

```
case 3:
display(queue);
break;


}
}while (ch != 0);
}

void insertq(int queue[], int item)
{
if ((front == 0 && rear == MAX - 1) || (front == rear + 1))
{
printf("QUEUE OVER-FLOW");
return;
}
else if (rear ==  - 1)
{
rear++;
front++;
}
else if (rear == MAX - 1 && front > 0)
{
rear = 0;
}
else
{
rear++;
}
queue[rear] = item;
}

void display(int queue[])
{
int i;
printf("\n");
if (front > rear)
{
for (i = front; i < MAX; i++)
{
printf("%d ", queue[i]);
}
```

```
for (i = 0; i <= rear; i++)
printf("%d ", queue[i]);
}
else
{
for (i = front; i <= rear; i++)
printf("%d ", queue[i]);
}
}

void deleteq(int queue[])
{
if (front ==  - 1)
{
printf("QUEUE UNDER-FLOW ");
}
else if (front == rear)
{
printf("\n %d REMOVED", queue[front]);
front =  - 1;
rear =  - 1;
}
else
{
printf("\n %d REMOVED", queue[front]);
front++;
}
}
```

OUTPUT

```
 CIRCULAR QUEUE CHOICES:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
0. EXIT
PRESS THE CHOICE: 1
THE NUMBER TO BE ENTERED: 54
 CIRCULAR QUEUE CHOICES:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
0. EXIT
PRESS THE CHOICE:3
54
```

```
 CIRCULAR QUEUE CHOICES:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
0. EXIT
PRESS THE CHOICE: 3
54
CIRCULAR QUEUE CHOICES:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
0. EXIT
PRESS THE CHOICE : 2
  54 REMOVED
 CIRCULAR QUEUE CHOICES:
 1. ENQUEUE
  2. DEQUEUE
 3. DISPLAY
0. EXIT
 PRESS THE CHOICE: 0
```

41. Write a C program to create a single linked list with 5 nodes. (5 integers are taken from user input) and display the linked-list elements.

```
 #include <stdio.h>
#include <stdlib.h>
struct node
{
int num;
struct node *nextptr;
}*stnode;
void createNodeList(int n);
int NodeCount();
void displayList();
int main()
{
int n,totalNode;
printf("\n\n CREATE A SINGLE LINKED LIST THEN COUNT THE TOTAL NODES:\n");
printf("-------------------------------------------------------------\n");
printf(" ENTER THE VALUE OF NODES: ");
scanf("%d", &n);
createNodeList(n);
printf("\n VALUES GIVEN IN THE SLL ARE : \n");
displayList();
```

```
totalNode = NodeCount();
printf("\n TOTAL NUMBER OF NODES = %d\n", totalNode);
return 0;
}

void createNodeList(int n)
{
struct node *fnNode, *tmp;
int num, i;
stnode = (struct node *)malloc(sizeof(struct node));
if(stnode == NULL)
{
printf(" MEMORY CAN'T BE GIVEN.");
}
else
{
printf(" GIVEN DATA FOR NODE 1 : ");
scanf("%d", &num);
stnode-> num = num;
stnode-> nextptr = NULL;
tmp = stnode;
for(i=2; i<=n; i++)
{
fnNode = (struct node *)malloc(sizeof(struct node));
if(fnNode == NULL)
{
printf(" MEMORY CAN'T BE GIVEN.");
break;
}
else
{
printf(" GIVEN DATA FOR NODE %d : ", i);
scanf(" %d", &num);
fnNode->num = num;
fnNode->nextptr = NULL;
tmp->nextptr = fnNode;
tmp = tmp->nextptr;
}
}
}
}

int NodeCount()
```

```
{
int ctr = 0;
struct node *tmp;
tmp = stnode;
while(tmp != NULL)
{
ctr++;
tmp = tmp->nextptr;
}
return ctr;
}

void displayList()
{
struct node *tmp;
if(stnode == NULL)
{
printf(" NO DATA FOUND IN THE LIST.");
}
else
{
tmp = stnode;
while(tmp != NULL)
{
printf(" DATA = %d\n", tmp->num);
tmp = tmp->nextptr;
}
}
}
```

OUTPUT

```
CREATE A SINGLE LINKED LIST THEN COUNT THE TOTAL NODES:
------------------------------------------------------------
 ENTER THE VALUE OF NODES: 3
 GIVEN DATA FOR NODE 1 : 10
 GIVEN DATA FOR NODE 2 : 20
 GIVEN DATA FOR NODE 3 : 30

  VALUES GIVEN IN THE SLL ARE :
 DATA = 10
 DATA = 20
 DATA = 30
  TOTAL NUMBER OF NODES = 3
```

42. Write a C program to search an element in a singly-linked

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

struct node
{
int num;
struct node *nextptr;
}

stnode, *ennode;

int FindElement(int);
void main()
{
int n,i,FindElem,FindPlc;
stnode.nextptr=NULL;
ennode=&stnode;
printf("\n\n LINKED LIST : SEARCH AN ELEMENT IN SLL :\n");
printf("-----------------------------------------\n");

printf(" ENTER THE VALUE OF NODES : ");
scanf("%d", &n);
printf("\n");
for(i=0;i< n;i++)
{
ennode->nextptr=(struct node *)malloc(sizeof(struct node));
printf(" INPUT DATA FOR NODE %d : ",i+1);
scanf("%d",&ennode->num);
ennode=ennode->nextptr;
}
ennode->nextptr=NULL;
printf("\n ENTERED ELEMENTS IN THE SLL ARE :\n");
ennode=&stnode;
while(ennode->nextptr!=NULL)
{
printf(" DATA = %d\n",ennode->num);
ennode=ennode->nextptr;
}

printf("\n");
printf(" TYPE THE ELEMENT TO BE SEARCHED : ");
scanf("%d",&FindElem);
```

```
FindPlc=FindElement(FindElem);
if(FindPlc<=n)
printf(" ELEMENT FOUND AT NODE %d \n\n",FindPlc);
else
printf(" ELEMENT DOES'NT EXISTS IN LINKED LIST.\n\n");
}

int FindElement(int FindElem)
{
int ctr=1;
ennode=&stnode;
while(ennode->nextptr!=NULL)
{
if(ennode->num==FindElem)
break;
else
ctr++;
ennode=ennode->nextptr;
}
return ctr;
}list.
```

OUTPUT

```
  LINKED LIST : SEARCH AN ELEMENT IN SLL :
------------------------------------------
 ENTER THE VALUE OF NODES : 2

 INPUT DATA FOR NODE 1 : 45
 INPUT DATA FOR NODE 2 : 789
 ENTERED ELEMENTS IN THE SLL ARE :
 DATA = 45
 DATA = 789
  TYPE THE ELEMENT TO BE SEARCHED : 45
 ELEMENT FOUND AT NODE 1
```

44. Write a C program to create a doubly linked list with 5 nodes.

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

struct node {
int num;
struct node * preptr;
struct node * nextptr;
}*stnode, *ennode;
```

```
void DlListcreation(int n);
void displayDlList();

int main()
{
int n;
stnode = NULL;
ennode = NULL;
printf("\n\n DOUBLY LINKED LIST : GENERATE AND DISPLAY A DLL :\n");
printf("-----------------------------------------------------\n");

printf(" ENTER THE NUMBER OF NODES : ");
scanf("%d", &n);

DlListcreation(n);
displayDlList();
return 0;
}

void DlListcreation(int n)
{
int i, num;
struct node *fnNode;

if(n >= 1)
{
stnode = (struct node *)malloc(sizeof(struct node));

if(stnode != NULL)
{
printf(" DATA ENTERED FOR NODE 1 : ");
scanf("%d", &num);

stnode->num = num;
stnode->preptr = NULL;
stnode->nextptr = NULL;
ennode = stnode;

for(i=2; i<=n; i++)
{
fnNode = (struct node *)malloc(sizeof(struct node));
```

```
if(fnNode != NULL)
{
printf(" DATA ENTERED FOR NODE %d : ", i);
scanf("%d", &num);
fnNode->num = num;
fnNode->preptr = ennode;
fnNode->nextptr = NULL;

ennode->nextptr = fnNode;
ennode = fnNode;
}
else
{
printf(" MEMORY CAN'T BE GIVEN.");
break;
}
}
}
else
{
printf(" MEMORY CAN'T BE GIVEN.");
}
}
}
void displayDlList()
{
struct node * tmp;
int n = 1;
if(stnode == NULL)
{
printf(" NO DATA FOUND IN THE LIST.");
}
else
{
tmp = stnode;
printf("\n\n DATA ENTERED ON THE LIST:\n");

while(tmp != NULL)
{
printf(" NODE %d : %d\n", n, tmp->num);
n++;
tmp = tmp->nextptr;
}
```

```
}
}
```

OUTPUT

```
 DOUBLY LINKED LIST : GENERATE AND DISPLAY A DLL :
--------------------------------------------------
 ENTER THE NUMBER OF NODES : 4
 DATA ENTERED FOR NODE 1 : 10
 DATA ENTERED FOR NODE 2 : 20
 DATA ENTERED FOR NODE 3 : 30
 DATA ENTERED FOR NODE 4 : 40
  DATA ENTERED ON THE LIST:
 NODE 1 : 10
 NODE 2 : 20
 NODE 3 : 30
 NODE 4 : 40
```

45. Write a C program to create a circular linked list with 5 nodes.

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

struct node {
int num;
struct node * nextptr;
}*stnode;


void ClListcreation(int n);
void displayClList();

int main()
{
int n;
stnode = NULL;
printf("\n\n CIRCULAR LINKED LIST : GENERATE AND DISPLAY A CLL:\n");
printf("-------------------------------------------------------\n");

printf(" TYPE THE NUMBER OF NODES : ");
scanf("%d", &n);

ClListcreation(n);
displayClList();
return 0;
}
```

```
void ClListcreation(int n)
{
int i, num;
struct node *preptr, *newnode;

if(n >= 1)
{
stnode = (struct node *)malloc(sizeof(struct node));

printf(" ENTERED DATA FOR NODE 1 : ");
scanf("%d", &num);
stnode->num = num;
stnode->nextptr = NULL;
preptr = stnode;
for(i=2; i<=n; i++)
{
newnode = (struct node *)malloc(sizeof(struct node));
printf(" ENTERED DATA FOR NODE %d : ", i);
scanf("%d", &num);
newnode->num = num;
newnode->nextptr = NULL;
preptr->nextptr = newnode;
preptr = newnode;
}
preptr->nextptr = stnode;
}
}

void displayClList()
{
struct node *tmp;
int n = 1;

if(stnode == NULL)
{
printf(" NONE DATA WAS FOUND IN THE LIST.");
}
else
{
tmp = stnode;
printf("\n\n DATA ALLOCATED IN THE LIST:\n");
```

```
do {
printf(" DATA %d = %d\n", n, tmp->num);
tmp = tmp->nextptr;
n++;
}while(tmp != stnode);
}
}
```

OUTPUT

```
 CIRCULAR LINKED LIST : GENERATE AND DISPLAY A CLL:
-----------------------------------------------------
 TYPE THE NUMBER OF NODES : 3
 ENTERED DATA FOR NODE 1 :20
ENTERED DATA FOR NODE 2 : 40
 ENTERED DATA FOR NODE 3 : 60
 DATA ALLOCATED IN THE LIST:
 DATA 1 = 20
 DATA 2 = 40
 DATA 3 = 60
```

46. Write a C program to implement the stack using linked list.

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

struct node
{
int info;
struct node *ptr;
}*top,*top1,*temp;

int topelement();
void push(int data);
void pop();
void empty();
void display();
void destroy();
void stack_count();
void create();

int count = 0;

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

```
printf("\n 1 - PUSH THE ELEMENT");
printf("\n 2 - POP THE ELEMENT");
printf("\n 3 - SHOW THE TOP ELEMENT");
printf("\n 4 - EMPTY THE LIST");
printf("\n 5 - EXIT");
printf("\n 6 - DISPLAY THE LIST");
printf("\n 7 - COUNT THE ELEMENT IN STACK");
printf("\n 8 - DESTROY THE STACK");

create();

while (1)
{
printf("\n TYPE YOUR CHOICE: ");
scanf("%d", &ch);

switch (ch)
{
case 1:
printf("ENTER THE ELEMENT: ");
scanf("%d", &no);
push(no);
break;

case 2:
pop();
break;

case 3:
if (top == NULL)
printf("THERE IS NO ELEMENT IN THE STACK");
else
{
e = topelement();
printf("\n ELEMENT IN THE TOP: %d", e);
}
break;

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

```
case 5:
exit(0);

case 6:
display();
break;

case 7:
stack_count();
break;

case 8:
destroy();
break;

default :
printf(" WRONG CHOICE,ENTER THE CORRECT CHOICE");
break;
}
}
}



void create()
{
top = NULL;
}


void stack_count()
{
printf("\n TOTAL ELEMENTS IN STACK: %d", count);
}


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;
}
count++;
}

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");
return;
}
else
top1 = top1->ptr;
printf("\n POPPED ELEMENT : %d", top->info);
free(top);
top = top1;
```

```
count--;
}



int topelement()
{
   return(top->info);
}


void empty()
{
if (top == NULL)
printf("\n STACK IS EMPTY");
else
printf("\n STACK IS NOT EMPTY WITH %d ELEMENTS", count);
}


void destroy()
{
top1 = top;

while (top1 != NULL)
{
top1 = top->ptr;
free(top);
top = top1;
top1 = top1->ptr;
}
free(top1);
top = NULL;
printf("\n  STACK ELEMENTS ARE DESTROYED");
count = 0;
}
```

OUTPUT

```
 1 - PUSH THE ELEMENT
 2 - POP THE ELEMENT
 3 - SHOW THE TOP ELEMENT
 4 - EMPTY THE LIST
 5 - EXIT
 6 - DISPLAY THE LIST
```

```
 7 - COUNT THE ELEMENT IN STACK
 8 - DESTROY THE STACK
 TYPE YOUR CHOICE: 1
ENTER THE ELEMENT: 43

 TYPE YOUR CHOICE: 1
ENTER THE ELEMENT: 52

 TYPE YOUR CHOICE: 4

 STACK IS NOT EMPTY WITH 2 ELEMENTS
 TYPE YOUR CHOICE: 1
ENTER THE ELEMENT: 65

 TYPE YOUR CHOICE: 8

  STACK ELEMENTS ARE DESTROYED
 TYPE YOUR CHOICE: 5
```

47. Write a C program to implement the queue using a linked list.

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

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

int frontelement();
void enq(int data);
void deq();
void empty();
void display();
void create();
void queuesize();

int count = 0;

void main()
{
int no, ch, e;
printf("\n------SELECT--------");
```

```
printf("\n 1 - ENQUEUE");
printf("\n 2 - DEQUEUE");
printf("\n 3 - FIRST ELEMENT");
printf("\n 4 - EMPTY");
printf("\n 5 - EXIT");
printf("\n 6 - DISPLAY THE LIST");
printf("\n 7 - SIZE OF THE QUEUE");
create();
while (1)
{
printf("\n ENTER THE CHOICE : ");
scanf("%d", &ch);
switch (ch)
{

case 1:
printf("TYPE THE DATA : ");
scanf("%d", &no);
enq(no);
break;

case 2:
deq();
break;

case 3:
e = frontelement();
if (e != 0)
printf("FIRST ELEMENT : %d", e);
else
printf("\n ABSENCE OF FIRST ELEMENT IN QUEUE");
break;

case 4:
empty();
break;

case 5:
exit(0);

case 6:
display();
break;
```

```
case 7:
queuesize();
break;

default:
printf("WRONG CHOICE,ENTER THE CORRECT CHOICE");
break;
}
}
}


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

void queuesize()
{
printf("\n SIZE OF QUEUE : %d", count);
}

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;
}
count++;
}
```

```
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 ERROR");
return;
}
else
if (front1->ptr != NULL)
{
front1 = front1->ptr;
printf("\n DEQUED ELEMENT : %d", front->info);
free(front);
front = front1;
}
else
{
printf("\n DEQUED ELEMENT : %d", front->info);
free(front);
front = NULL;
rear = NULL;
}
count--;
```

```
}

int frontelement()
{
if ((front != NULL) && (rear != NULL))
return(front->info);
else
return 0;
}

void empty()
{
if ((front == NULL) && (rear == NULL))
printf("\n QUEUE IS EMPTY");
else
printf("QUEUE IS NOT EMPTY");
}
```

OUTPUT

```
 ------SELECT--------
 1 - ENQUEUE
 2 - DEQUEUE
 3 - FIRST ELEMENT
 4 - EMPTY
 5 - EXIT
 6 - DISPLAY THE LIST
 7 - SIZE OF THE QUEUE
 ENTER THE CHOICE : 1
TYPE THE DATA : 75
 ENTER THE CHOICE : 1
TYPE THE DATA : 54
 ENTER THE CHOICE : 3
FIRST ELEMENT : 75
 ENTER THE CHOICE : 5
```