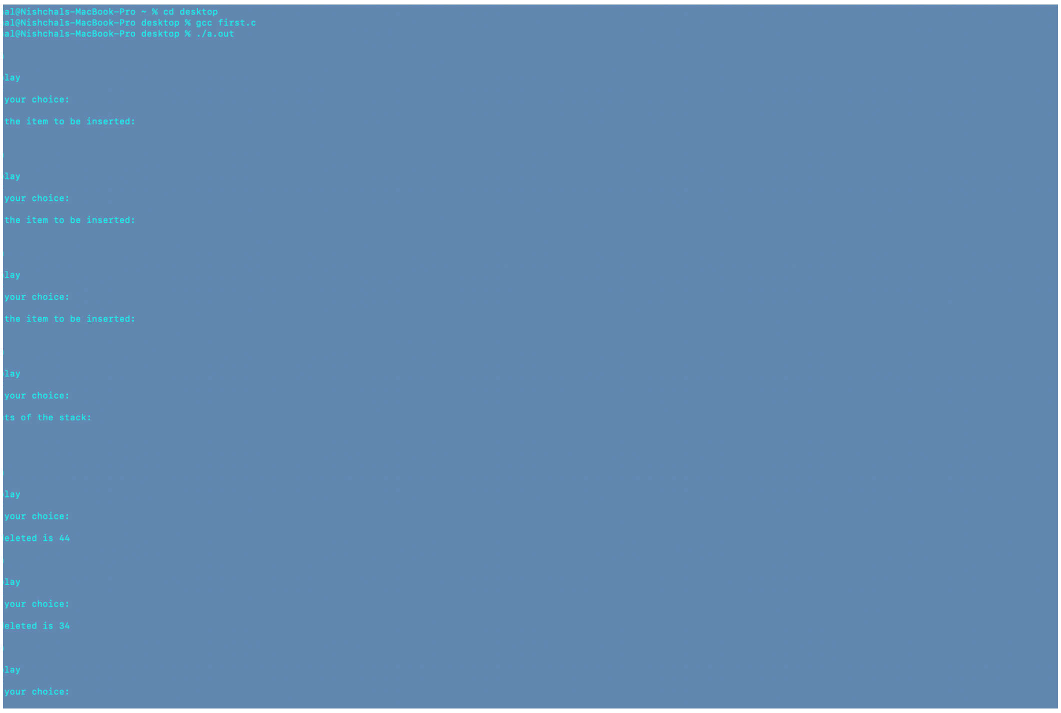
Name: NISHCHAL NANDAGOPAL
USN : 1BM19CS105
SEC: 3 B
SUBJECT: DATA STRUCTURES LAB
ACADEMIC YEAR: 2020

# Lab Program 1:

Write a program to simulate the working of stack using an array with the following : a) Push b) Pop c) Display The program should print appropriate messages for stack overflow, stack underflow

```c
#include<stdio.h>
#include<stdlib.h>
#define STACK_SIZE 3
int top=-1;
int s[10];
int item;
void push()
{
if(top>=STACK_SIZE-1)
{
printf("Stack Overflow\n");
return;
}
top=top+1;
s[top]=item;
}
int pop()
{
if(top==-1)
return -1;
return s[top--];
}
void display()
{
int i;
if(top==-1)
{
printf("Stack is empty\n");
return;
}
printf("Contents of the stack:\n");
for(i=0;i<=top;i++)
{
printf("%d\n",s[i]);
}
}
void main()
{
int item_deleted;
int choice;
for(;;)
{
printf("\n1:Push\n2:Pop\n3:Display\n4:Exit\n");
printf("Enter your choice:\n");
scanf("%d",&choice);
switch(choice)
{
case 1:
printf("Enter the item to be inserted:\n");
scanf("%d",&item);
push();
break;
case 2:
item_deleted=pop();
if(item_deleted==-1)
printf("stack is empty\n");
else
printf("item deleted is %d\n",item_deleted);
break;
case 3:
display();
break;
case 4:exit(0);
}
}
}
```



```
al@Nishchals-MacBook-Pro ~ % cd desktop
al@Nishchals-MacBook-Pro desktop % gcc first.c
al@Nishchals-MacBook-Pro desktop % ./a.out



lay

your choice:

the item to be inserted:




lay

your choice:

the item to be inserted:




lay

your choice:

the item to be inserted:




lay

your choice:

ts of the stack:






lay

your choice:

eleted is 44



lay

your choice:

eleted is 34



lay

your choice:
```

TYPE TO ENTER A CAPTION.

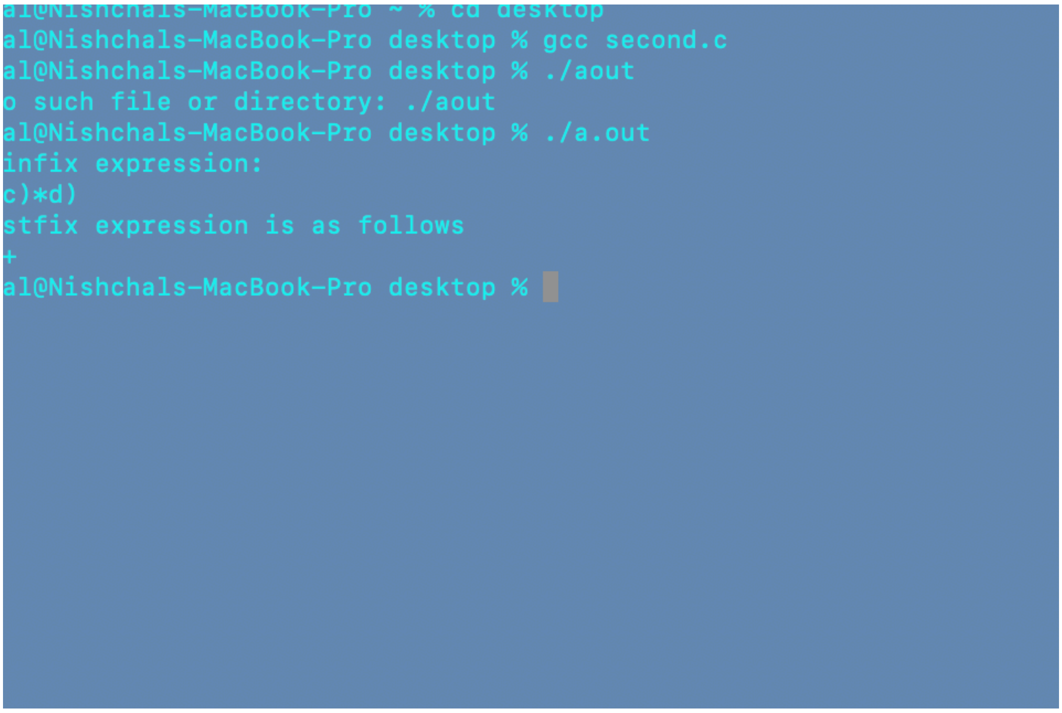
## Lab Program 2:

WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide) 1

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

int F(char symbol) {
switch(symbol) {
case '+':
case '-': return 2;
case '*':
case '/': return 4;
case '^':
case '$': return 5;
case '(': return 0;
case '#': return -1;
default : return 8;
}
}

int G(char symbol) {
switch(symbol) {
case '+':
case '-': return 1;
case '*':
case '/': return 3;
case '^':
case '$': return 6;
case '(': return 9;
case ')': return 0;
default : return 7;
}
}

```

```
void infix_postfix(char infix[], char postfix[]) {
int top,i,j;
char s[30], symbol;
top =-1;
s[++top] = '#';
j=0;

for(i=0;i<strlen(infix);i++) {
symbol = infix[i];
while(F(s[top])>G(symbol)) {
postfix[j] = s[top--];
j++;
}
if(F(s[top]) != G(symbol)) {
s[++top] = symbol;
}
else {
top--;
}
}
while(s[top]!='#') {
postfix[j++]=s[top--];
}
postfix[j] = '\0';
}

void main() {
char infix[20];
char postfix[20];
printf("Enetr infix expression:\n");
scanf("%s",infix);
infix_postfix(infix,postfix);
printf("The postfix expression is as follows \n");
printf("%s\n",postfix);
```

| | |
|---|---|
| | } |



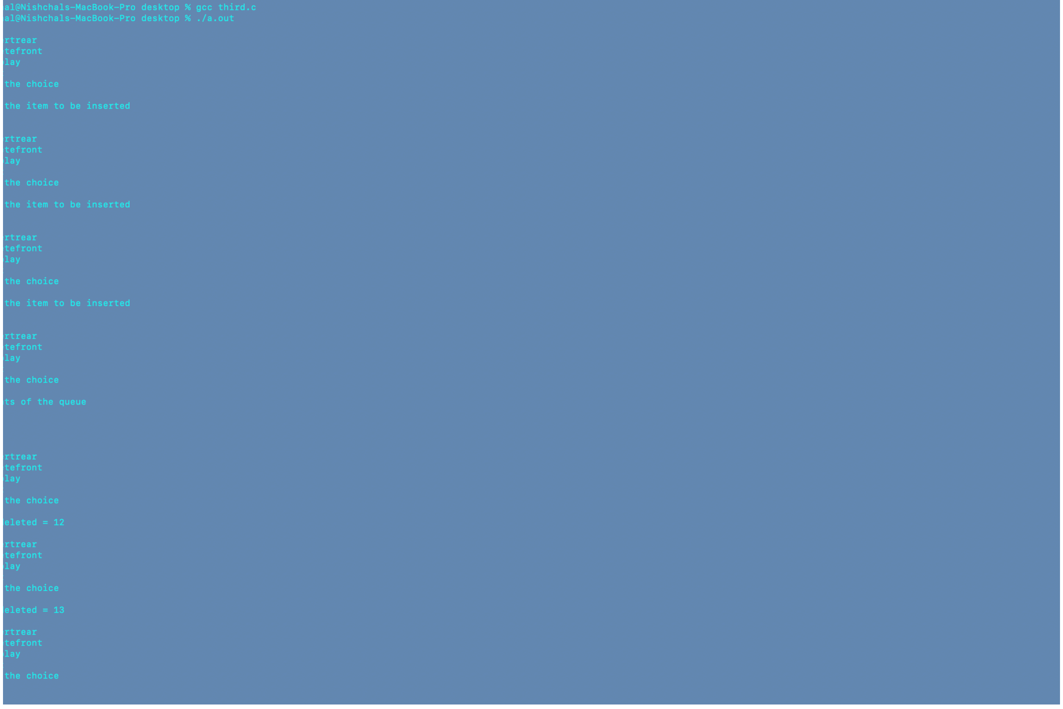
```
al@Nishchals-MacBook-Pro ~ % cd desktop
al@Nishchals-MacBook-Pro desktop % gcc second.c
al@Nishchals-MacBook-Pro desktop % ./aout
o such file or directory: ./aout
al@Nishchals-MacBook-Pro desktop % ./a.out
infix expression:
c)*d)
stfix expression is as follows
+
al@Nishchals-MacBook-Pro desktop %
```

**TYPE TO ENTER A CAPTION.**

# Lab Program 3:

WAP to simulate the working of a queue of integers using an array. Provide the following operations a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions

```
#include<stdio.h>
#include<stdlib.h>
//#include<conio.h>
//#include<process.h>
#define QUE_SIZE 3
int item, front = 0, rear =
-1, q[10];

void insertRear() {
if(rear == QUE_SIZE-1){
printf("Queue Overflow \n");

}
rear = rear+1;
q[rear] = item;
}

int deleteFront() {
if(front>rear){
front = 0;
rear = -1;
return -1;
}
return q[front++];
}

void displayQ() {
int i;
if(front>rear){
printf("queue is empty\n");
}
printf("Contents of the queue\n");
for(i=front;i<=rear;i++){
printf("%d\n",q[i]);
}
}

int main() {
int choice = 0;
for(;;){
printf("\n1:insertrear\n2:deletefront\n3:display\n4:exit\n");
printf("enter the choice\n");
scanf("%d",&choice);

switch(choice){
case 1: printf("Enter the item to be inserted\n");
scanf("%d",&item);
insertRear();
break;
case 2: item = deleteFront();
if(item == -1)
printf("queue is empty\n");
else
printf("item deleted = %d\n", item);
break;
case 3: displayQ();
break;
default:exit(0);

}
}
}
```



```
al@Nishchals-MacBook-Pro desktop % gcc third.c
al@Nishchals-MacBook-Pro desktop % ./a.out

rtrear
tefront
lay

the choice

the item to be inserted


rtrear
tefront
lay

the choice

the item to be inserted


rtrear
tefront
lay

the choice

the item to be inserted


rtrear
tefront
lay

the choice

ts of the queue




rtrear
tefront
lay

the choice

eleted = 12

rtrear
tefront
lay

the choice

eleted = 13

rtrear
tefront
lay

the choice
```

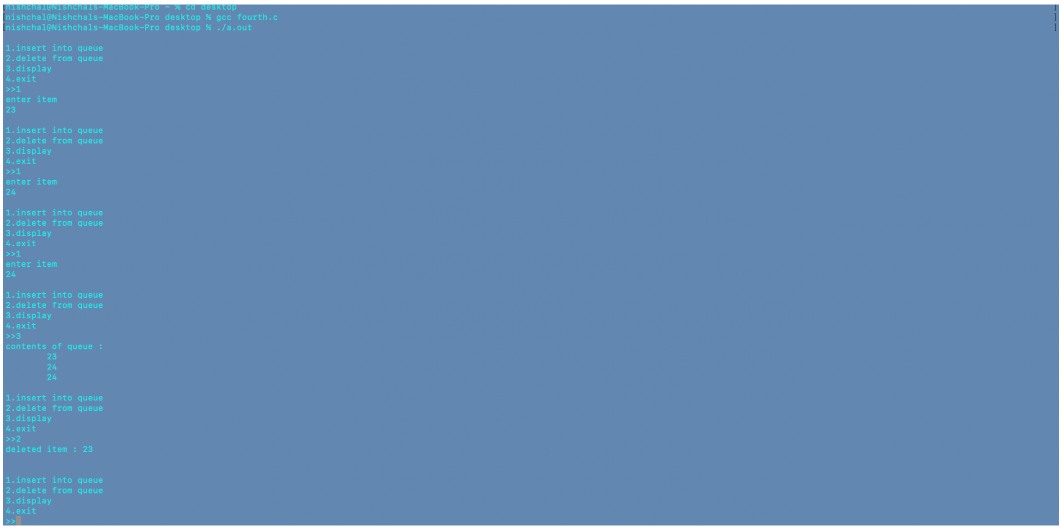
**TYPE TO ENTER A CAPTION.**

# Lab program 4:

WAP to simulate the working of a Circular queue of integers using an array. Provide the following operations a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions

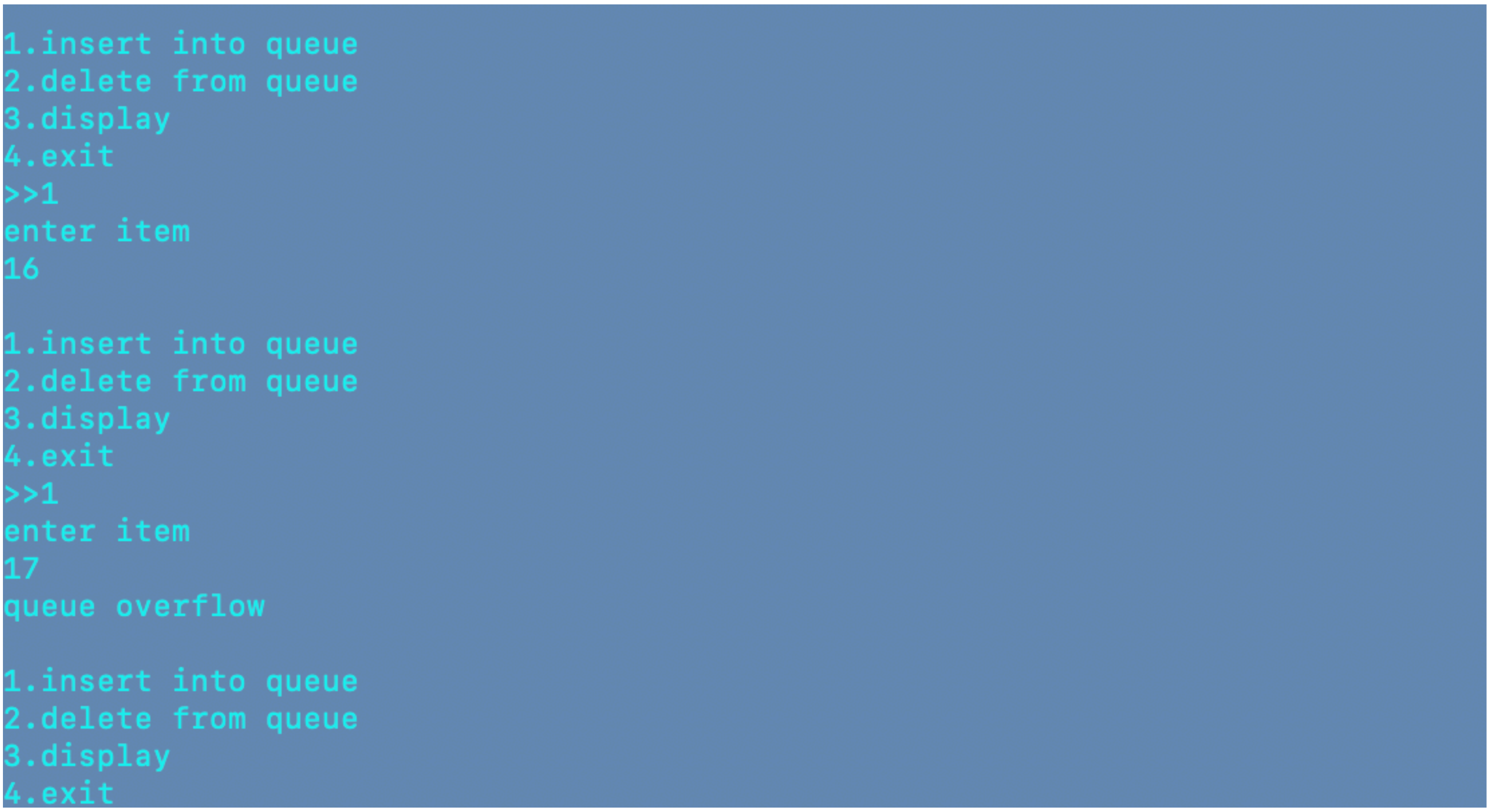
```
#include<stdio.h>
#include<stdlib.h>
int item,front=0,rear=-1,q[10],count=0,qSize=5;
void insert()
{
if(count==qSize)
{
printf("queue overflow \n");
return;
}
rear=(rear+1)%qSize;
q[rear]=item;
count++;
}
int delete()
{
if(count==0)
{
return(-1);
```

```c
}
item=q[front];
front=(front+1)%qSize;
count--;
return(item);
}
void display()
{
if(count==0)
{
printf("queue is empty \n");
return;
}
printf("contents of queue :\n");
int f=front;
for(int i=1;i<=count;i++)
{
printf("\t%d \n",q[f]);
f=(f+1)%qSize;
}
}
int main()
{
int n;
for(;;)
{
printf("\n1.insert into queue \n2.delete from queue \n3.display \n4.exit\n>>");
scanf("%d",&n);
switch(n)
{
case 1:printf("enter item \n");
scanf("%d",&item);
insert();
break;
case 2:item=delete();
if(item==-1)
printf("queue is empty\n");
else
printf("deleted item : %d\n\n",item);
break;
case 3:display();
break;
default:exit(0);
}
}
}
```



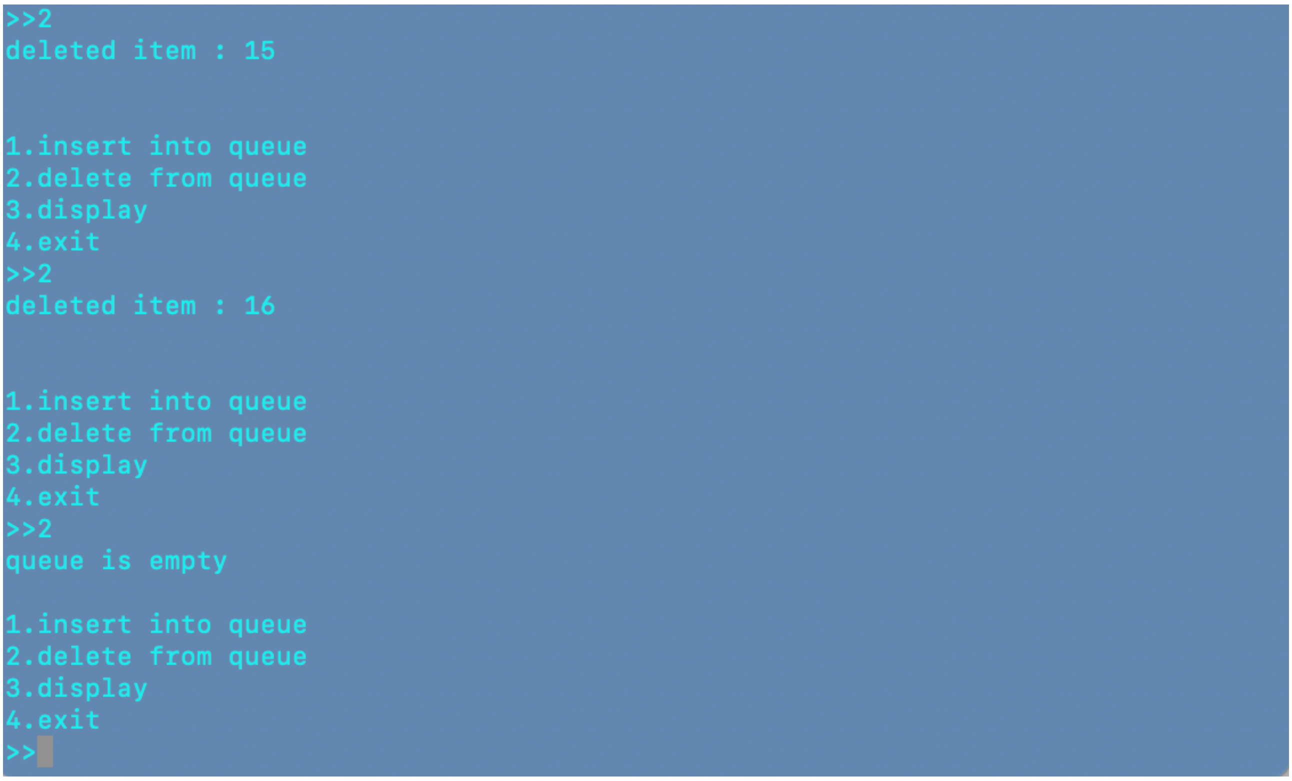
```
nishchal@Nishchals-MacBook-Pro ~ % cd desktop
nishchal@Nishchals-MacBook-Pro desktop % gcc fourth.c
nishchal@Nishchals-MacBook-Pro desktop % ./a.out

1.insert into queue
2.delete from queue
3.display
4.exit
>>1
enter item
23

1.insert into queue
2.delete from queue
3.display
4.exit
>>1
enter item
24

1.insert into queue
2.delete from queue
3.display
4.exit
>>1
enter item
24

1.insert into queue
2.delete from queue
3.display
4.exit
>>3
contents of queue :
        23
        24
        24

1.insert into queue
2.delete from queue
3.display
4.exit
>>2
deleted item : 23


1.insert into queue
2.delete from queue
3.display
4.exit
>>
```

**TYPE TO ENTER A CAPTION.**



```
1.insert into queue
2.delete from queue
3.display
4.exit
>>1
enter item
16

1.insert into queue
2.delete from queue
3.display
4.exit
>>1
enter item
17
queue overflow

1.insert into queue
2.delete from queue
3.display
4.exit
```

TYPE TO ENTER A CAPTION.



```
>>2
deleted item : 15


1.insert into queue
2.delete from queue
3.display
4.exit
>>2
deleted item : 16


1.insert into queue
2.delete from queue
3.display
4.exit
>>2
queue is empty

1.insert into queue
2.delete from queue
3.display
4.exit
>>
```

## Lab Program 5:

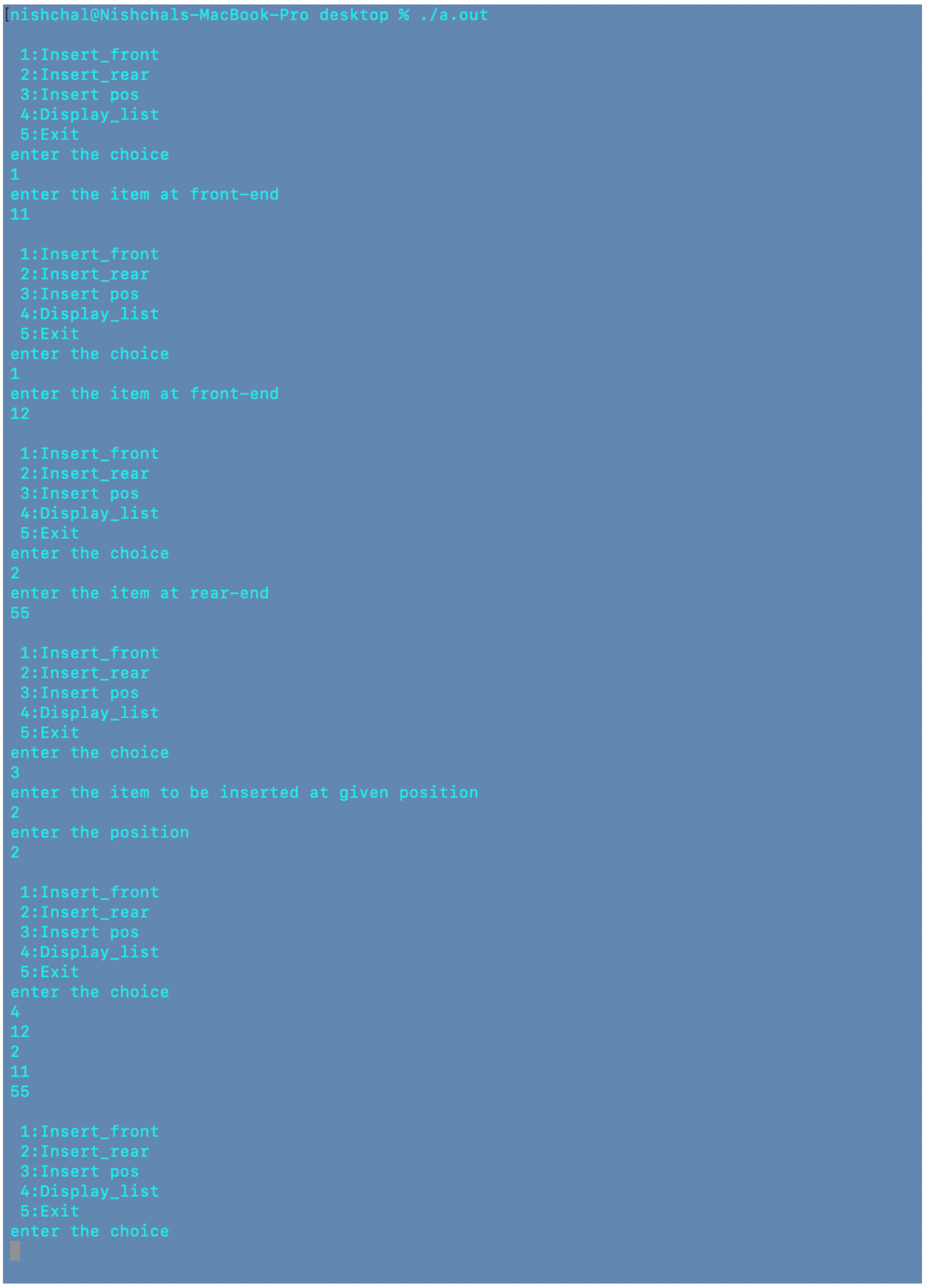
WAP to Implement Singly Linked List with following operations a) Create a linked list. b) Insertion of a node at first position, at any position and at end of list. c) Display the contents of the linked list.

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

struct node
{
int info;
struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct
node));
if(x==NULL)
{
printf("mem full\n");
exit(0);
}
return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert_front(NODE
first,int item)
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
temp->link=first;
```

```
first=temp;
return first;
}

NODE insert_rear(NODE
first,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
cur=first;
while(cur->link!=NULL)
cur=cur->link;
cur->link=temp;
return first;
}
NODE insert_pos(int item,int
pos,NODE first)
{
NODE temp,cur,prev;
int count;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL&&pos==1)
{
return temp;
}
if(first==NULL)
{
printf("invalid position\n");
return first;
}
if(pos==1)
```

```
{
temp->link=first;
first=temp;
return temp;
}
count=1;
prev=NULL;
cur=first;
while(cur!=NULL&&count!=pos)
{
prev=cur;
cur=cur->link;
count++;
}
if(count==pos)
{
prev->link=temp;
temp->link=cur;
return first;
}
printf("invalid position\n");
return first;
}
void display(NODE first)
{
NODE temp;
if(first==NULL)
printf("list empty cannot display items\n");
for(temp=first;temp!=NULL;temp=temp->link)
{
printf("%d\n",temp->info);
}
}
int main()
{
```

```
int item,choice,pos;
NODE first=NULL;
for(;;)
{
printf("\n 1:Insert_front\n 2:Insert_rear\n 3:Insert pos\n 4:Display_list\n 5:Exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the item at front-end\n");
scanf("%d",&item);
first=insert_front(first,item);
break;
case 2:printf("enter the item at rear-end\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 3:printf("enter the item to be inserted at given position\n");
scanf("%d",&item);
printf("enter the position\n");
scanf("%d",&pos);
first=insert_pos(item,pos,first);
break;
case 4:display(first);
break;
default:exit(0);
break;
}
}
}
```

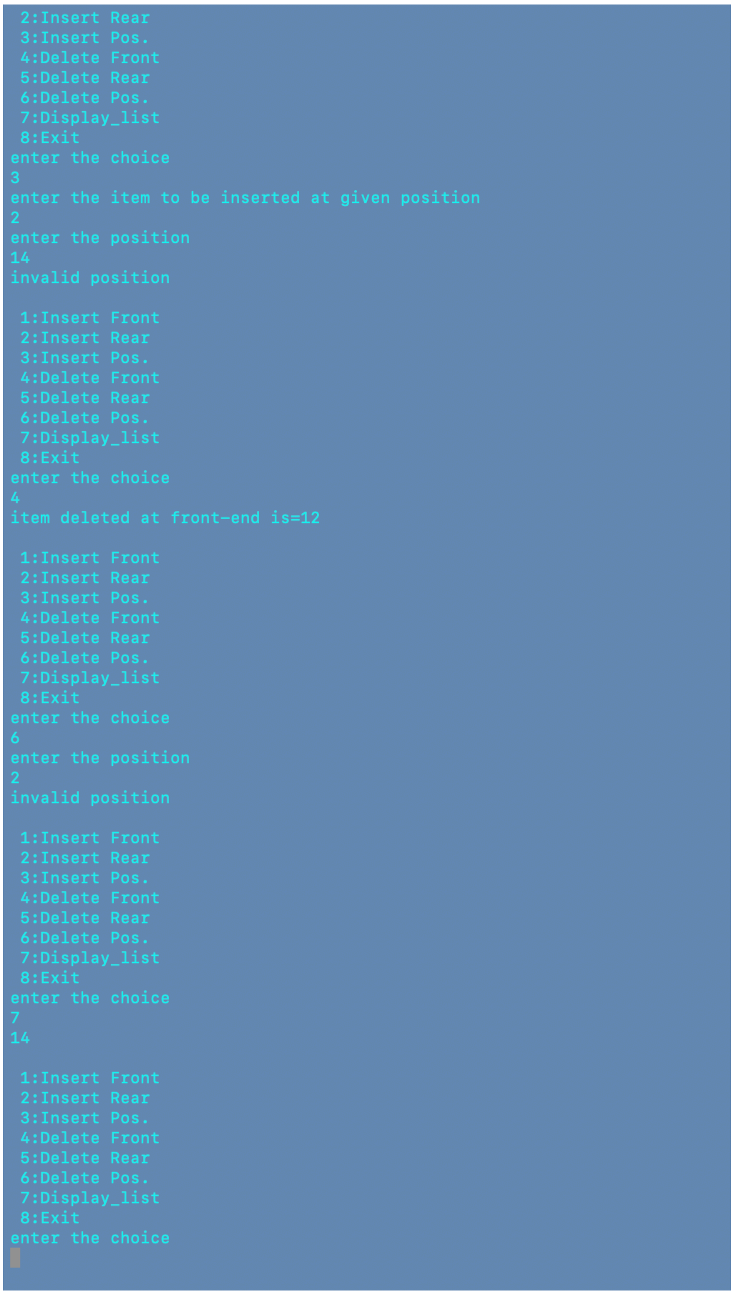


```
[nishchal@Nishchals-MacBook-Pro desktop % ./a.out

 1:Insert_front
 2:Insert_rear
 3:Insert pos
 4:Display_list
 5:Exit
enter the choice
1
enter the item at front-end
11

 1:Insert_front
 2:Insert_rear
 3:Insert pos
 4:Display_list
 5:Exit
enter the choice
1
enter the item at front-end
12

 1:Insert_front
 2:Insert_rear
 3:Insert pos
 4:Display_list
 5:Exit
enter the choice
2
enter the item at rear-end
55

 1:Insert_front
 2:Insert_rear
 3:Insert pos
 4:Display_list
 5:Exit
enter the choice
3
enter the item to be inserted at given position
2
enter the position
2

 1:Insert_front
 2:Insert_rear
 3:Insert pos
 4:Display_list
 5:Exit
enter the choice
4
12
2
11
55

 1:Insert_front
 2:Insert_rear
 3:Insert pos
 4:Display_list
 5:Exit
enter the choice
```

## Lab Program 6:

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>
#include<stdlib.h>

struct node
{
  int info;
  struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
  NODE x;
  x=(NODE)malloc(sizeof(struct node));
  if(x==NULL)
  {
    printf("mem full\n");
    exit(0);
  }
  return x;
}
void freenode(NODE x)
{
  free(x);
}
NODE insert_front(NODE first,int item)
{
  NODE temp;
  temp=getnode();
  temp->info=item;
  temp->link=NULL;
  if(first==NULL)
    return temp;
  temp->link=first;
  first=temp;
  return first;
}

NODE insert_rear(NODE first,int item)
{
  NODE temp,cur;
  temp=getnode();
  temp->info=item;
  temp->link=NULL;
  if(first==NULL)
    return temp;
  cur=first;
  while(cur->link!=NULL)
    cur=cur->link;
  cur->link=temp;
  return first;
}
NODE insert_pos(int item,int pos,NODE first)
{
  NODE temp,cur,prev;
  int count;
  temp=getnode();
  temp->info=item;
  temp->link=NULL;
  if(first==NULL&&pos==1)
  {
    return temp;
  }
  if(first==NULL)
  {
    printf("invalid position\n");
    return first;
  }
  if(pos==1)
  {
    temp->link=first;
    first=temp;
    return temp;
  }
  count=1;
  prev=NULL;
  cur=first;
  while(cur!=NULL&&count!=pos)
  {
    prev=cur;
    cur=cur->link;
    count++;
  }
  if(count==pos)
  {
    prev->link=temp;
    temp->link=cur;
    return first;
  }
  printf("invalid position\n");
  return first;
}

NODE delete_front(NODE first)
{
  NODE temp;
  if(first==NULL)
  {
    printf("list is empty cannot delete\n");
    return first;
  }
  temp=first;
  temp=temp->link;
  printf("item deleted at front-end is=%d\n",first->info);
  free(first);
  return temp;
}

NODE delete_rear(NODE first)
{
  NODE cur,prev;
  if(first==NULL)
  {
    printf("list is empty cannot delete\n");
    return first;
  }
  if(first->link==NULL)
  {
    printf("item deleted is %d\n",first->info);
    free(first);
    return NULL;
  }
  prev=NULL;
  cur=first;
  while(cur->link!=NULL)
  {
    prev=cur;
    cur=cur->link;
  }
  printf("iten deleted at rear-end is %d",cur->info);
  free(cur);
  prev->link=NULL;
  return first;
}

NODE delete_pos(int pos,NODE first)
{
  NODE cur;
  NODE prev;
  int count,flag=0;
  if(first==NULL || pos<0)
  {
    printf("invalid position\n");
    return NULL;
  }
  if(pos==1)
  {
    cur=first;
    first=first->link;
    freenode(cur);
    return first;
  }
  prev=NULL;
  cur=first;
  count=1;
  while(cur!=NULL)
  {
    if(count==pos){flag=1;break;}
    count++;
    prev=cur;
    cur=cur->link;
  }
  if(flag==0)
  {
    printf("invalid position\n");
    return first;
  }
  printf("item deleted at given position is %d\n",cur->info);
  prev->link=cur->link;
  freenode(cur);
  return first;
}


void display(NODE first)
{
  NODE temp;
  if(first==NULL)
    printf("list empty cannot display items\n");
  for(temp=first;temp!=NULL;temp=temp->link)
  {
    printf("%d\n",temp->info);
  }
}
int main()
{
  int item,choice,pos;
  NODE first=NULL;
  for(;;)
  {
    printf("\n 1:Insert Front\n 2:Insert Rear\n 3:Insert Pos.\n 4:Delete Front\n 5:Delete Rear\n 6:Delete Pos.\n 7:Display_list\n 8:Exit\n");
    printf("enter the choice\n");
    scanf("%d",&choice);
    switch(choice)
    {
      case 1:printf("enter the item at front-end\n");
             scanf("%d",&item);
             first=insert_front(first,item);
             break;
      case 2:printf("enter the item at rear-end\n");
             scanf("%d",&item);
             first=insert_rear(first,item);
             break;
      case 3:printf("enter the item to be inserted at given position\n");
             scanf("%d",&item);
             printf("enter the position\n");
             scanf("%d",&pos);
             first=insert_pos(item,pos,first);
             break;
      case 4:first=delete_front(first);
             break;
      case 5:first=delete_rear(first);
             break;
      case 6:printf("enter the position\n");
             scanf("%d",&pos);
             first=delete_pos(pos,first);
             break;
      case 7:display(first);
             break;
      default:exit(0);
              break;
    }
  }
}
```



```
 2:Insert Rear
 3:Insert Pos.
 4:Delete Front
 5:Delete Rear
 6:Delete Pos.
 7:Display_list
 8:Exit
enter the choice
3
enter the item to be inserted at given position
2
enter the position
14
invalid position

 1:Insert Front
 2:Insert Rear
 3:Insert Pos.
 4:Delete Front
 5:Delete Rear
 6:Delete Pos.
 7:Display_list
 8:Exit
enter the choice
4
item deleted at front-end is=12

 1:Insert Front
 2:Insert Rear
 3:Insert Pos.
 4:Delete Front
 5:Delete Rear
 6:Delete Pos.
 7:Display_list
 8:Exit
enter the choice
6
enter the position
2
invalid position

 1:Insert Front
 2:Insert Rear
 3:Insert Pos.
 4:Delete Front
 5:Delete Rear
 6:Delete Pos.
 7:Display_list
 8:Exit
enter the choice
7
14

 1:Insert Front
 2:Insert Rear
 3:Insert Pos.
 4:Delete Front
 5:Delete Rear
 6:Delete Pos.
 7:Display_list
 8:Exit
enter the choice
```

## Lab Program 7 and Lab Program 8:

WAP Implement Single Link List with following operations a) Sort the linked list. b) Reverse the linked list. c) Concatenation of two linked lists d) Stack and Queue Implementation

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int info;
struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("mem full\n");
exit(0);
}
return x;
}


NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
cur=first;
while(cur->link!=NULL)
cur=cur->link;
```

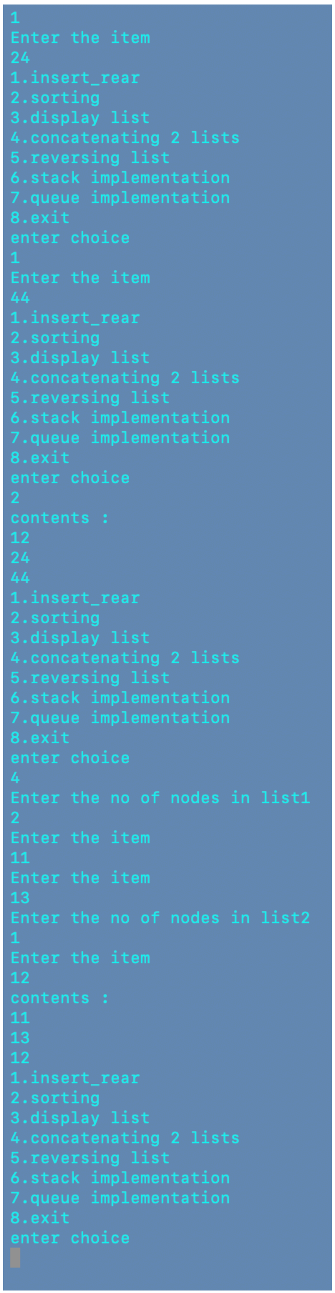
```
cur->link=temp;
return first;
}
void display(NODE first)
{
NODE temp;
if(first==NULL)
printf("list is empty");
printf("contents : \n");
for(temp=first;temp!=NULL;temp=temp->link)
{
printf("%d\n",temp->info);
}
}
NODE sort(NODE first)
{
int swapped;
NODE ptr1;
NODE lptr = NULL;
if (first == NULL)
return NULL;
do
{
swapped = 0;
ptr1 = first;

while (ptr1->link != lptr)
{
if (ptr1->info > ptr1->link->info)
{

int tem = ptr1->info;
ptr1->info = ptr1->link->info;
ptr1->link->info = tem;
swapped = 1;
}
ptr1 = ptr1->link;
}
lptr = ptr1;
} while (swapped);
}
NODE reverse(NODE first)
{
NODE cur,temp;
cur=NULL;
while(first!=NULL)
{
temp=first;
first=first->link;
temp->link=cur;
cur=temp;
}
return cur;
}

NODE concat(NODE first,NODE second)
{
NODE cur;
if(first==NULL)
return second;
if(second==NULL)
return first;
cur=first;
while(cur->link!=NULL)
cur=cur->link;
cur->link=second;
return first;
}

NODE delete_front(NODE first)
{
```

```
NODE temp;
if(first==NULL)
{
printf("list is empty \n");
return first;
}

temp=first->link;
printf("deleted item at front = %d\n ",first->info);
free(first);
return temp;
}

NODE delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("list is empty \n");
return first;
}
if(first->link==NULL)
{
printf("only one item in list and delete item  = %d ",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->link!=NULL)
{
prev=cur;
cur=cur->link;
}
```

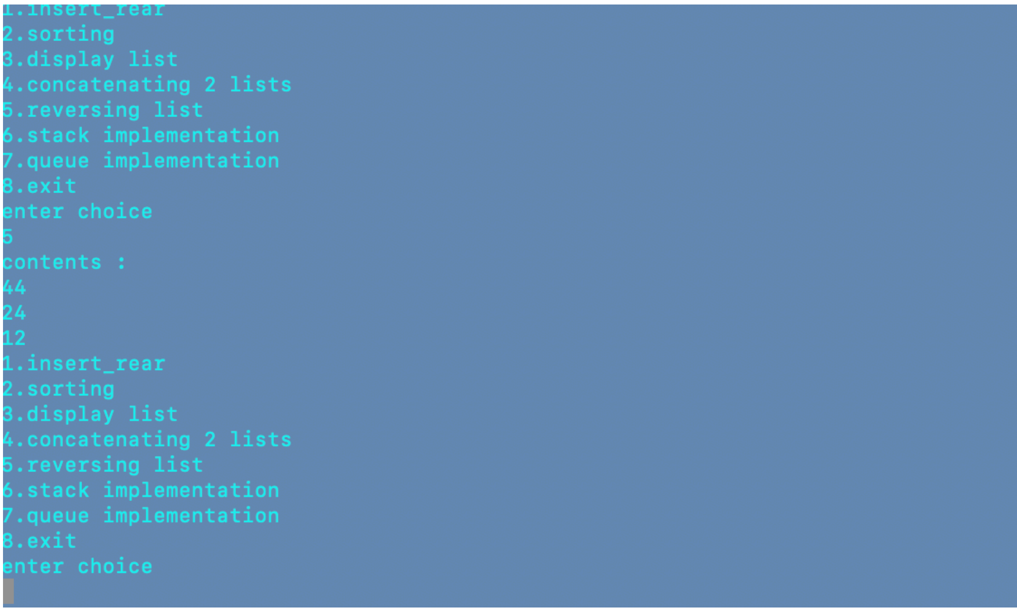
```
printf("deleted item at rear = %d \n ",cur->info);
free(cur);
prev->link=NULL;
return first;
}


void main()
{
int item,choice,ch,n;
NODE first=NULL,a,b;
NODE stack_first=NULL,queue_first=NULL;
for(;;)
{
printf("1.insert_rear\n2.sorting\n3.display list \n4.concatenating 2 lists \n5.reversing list \n6.stack implementation\n7.queue implementation\n8.exit\n");
printf("enter choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("Enter the item\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;

case 2:sort(first);
display(first);
break;
case 3:display(first);
break;
case 4:printf("Enter the no of nodes in list1\n");
```

```
scanf("%d",&n);
a=NULL;
for(int i=0;i<n;i++)
{
printf("Enter the item\n");
scanf("%d",&item);
a=insert_rear(a,item);
}
printf("Enter the no of nodes in list2\n");
scanf("%d",&n);
b=NULL;
for(int i=0;i<n;i++)
{
printf("Enter the item\n");
scanf("%d",&item);
b=insert_rear(b,item);
}
a=concat(a,b);
display(a);
break;
case 5:first=reverse(first);
display(first);
break;
case 6:printf("Stack\n");
for(;;)
{
printf("\n 1:Insert_rear\n 2:Delete_rear\n 3:Display_list\n 4:Exit\n");
printf("Enter the choice\n");
scanf("%d",&ch);
switch(ch)
{
case 1:printf("Enter the item at rear-end\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 2:first=delete_rear(first);
break;
case 3:display(first);
break;
default:ch=0;
}
if(ch==0)
break;
}
break;
case 7: printf("QUEUE\n");
for(;;)
{
printf("\n 1:Insert_rear\n 2:Delete_front\n 3:Display_list\n 4:Exit\n");
printf("Enter the choice\n");
scanf("%d",&ch);
switch(ch)
{
case 1:printf("Enter the item at rear-end\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 2:first=delete_front(first);
break;
case 3:display(first);
break;
default:ch=0;
}
if(ch==0)
break;
}
break;
default:exit(0);
```

| | |
|---|---|
| | } |



```
1
Enter the item
24
1.insert_rear
2.sorting
3.display list
4.concatenating 2 lists
5.reversing list
6.stack implementation
7.queue implementation
8.exit
enter choice
1
Enter the item
44
1.insert_rear
2.sorting
3.display list
4.concatenating 2 lists
5.reversing list
6.stack implementation
7.queue implementation
8.exit
enter choice
2
contents :
12
24
44
1.insert_rear
2.sorting
3.display list
4.concatenating 2 lists
5.reversing list
6.stack implementation
7.queue implementation
8.exit
enter choice
4
Enter the no of nodes in list1
2
Enter the item
11
Enter the item
13
Enter the no of nodes in list2
1
Enter the item
12
contents :
11
13
12
1.insert_rear
2.sorting
3.display list
4.concatenating 2 lists
5.reversing list
6.stack implementation
7.queue implementation
8.exit
enter choice
```



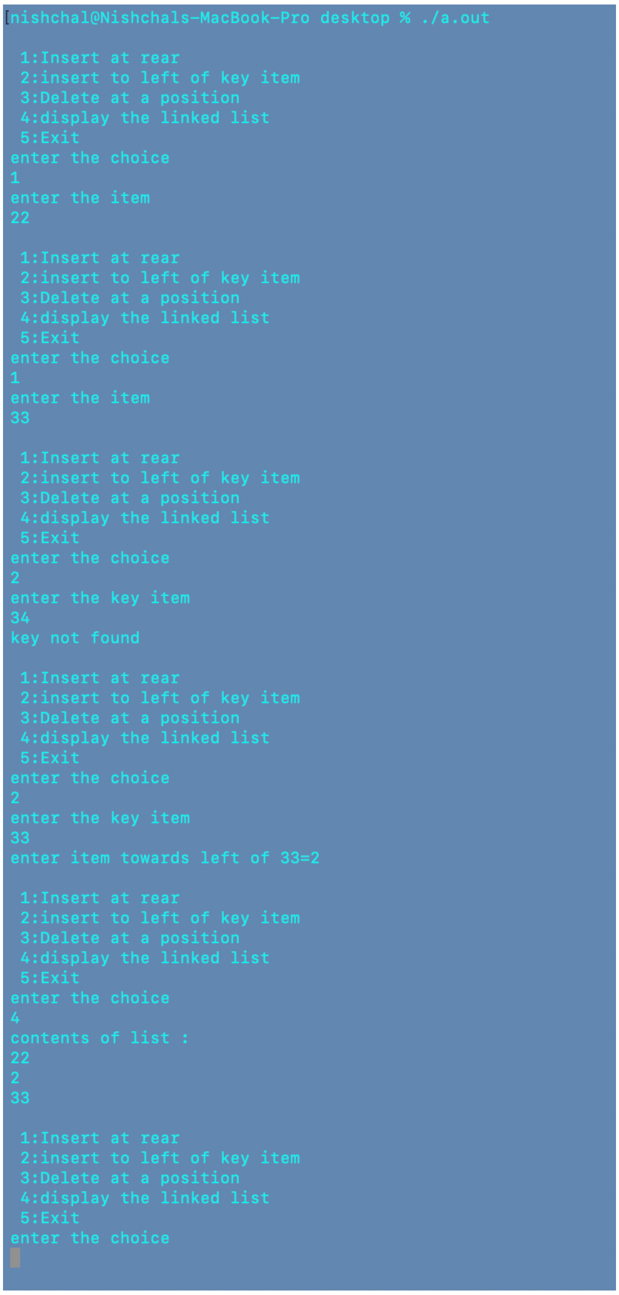
```
1.insert_rear
2.sorting
3.display list
4.concatenating 2 lists
5.reversing list
6.stack implementation
7.queue implementation
8.exit
enter choice
5
contents :
44
24
12
1.insert_rear
2.sorting
3.display list
4.concatenating 2 lists
5.reversing list
6.stack implementation
7.queue implementation
8.exit
enter choice
```

## Lab Program 9:

WAP Implement doubly link list with primitive operations a) Create a doubly linked list. b) Insert a new node to the left of the node. c) Delete the node based on a specific value d) Display the contents of the list

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

struct node
{
  int info;
  struct node *llink;
  struct node *rlink;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
 {
  printf("mem full\n");
  exit(0);
 }
 return x;
}

NODE insert_rear(int item,NODE head)
{
        NODE temp,cur;
        temp=getnode();
        temp->info=item;
        cur=head->llink;
        head->llink=temp;
        temp->rlink=head;
        temp->llink=cur;
        cur->rlink=temp;
        return head;
}

NODE insert_leftpos(int item,NODE head)
{
NODE temp,cur,prev;
if(head->rlink==head)
{
printf("list empty\n");
return head;
}
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)break;
cur=cur->rlink;
}
if(cur==head)
{
 printf("key not found\n");
 return head;
 }
 prev=cur->llink;
 printf("enter item towards left of %d=",item);
 temp=getnode();
 scanf("%d",&temp->info);
 prev->rlink=temp;
 temp->llink=prev;
 cur->llink=temp;
 temp->rlink=cur;
 return head;
}

NODE delete_position(int pos,NODE head)
{
	NODE p,q;
	int c=0;
	if(head==NULL)
	{
		printf("empty list \n");
		return head;
	}
	p=head;
	while((p->rlink!=NULL)&&(c!=pos))
	{
		q=p;
		p=p->rlink;
		c++;
	}
	if(c==pos)
	{
		printf("deleted item at %d = %d ",pos,p->info);
		q->rlink=p->rlink;
		if(p->rlink!=NULL)
			(p->rlink)->llink=q;
		free(p);
	}
	else
		printf("invalid position \n");
	return head;
}

void display(NODE head)
{
	if(head->rlink==head)
	{
		printf("empty list \n");
	}
	printf("contents of list : \n");
	NODE temp;
	temp=head->rlink;
	while(temp!=head)
	{
		printf("%d\n",temp->info);
		temp=temp->rlink;
	}
}

int main()
{
	NODE head;
int item, choice,pos;
head=getnode();
```

```
head->rlink=head;
head->llink=head;
for(;;)
{
printf("\n 1:Insert at rear\n 2:insert to left of key item \n 3:Delete at a position\n 4:display the linked
list \n 5:Exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
  case 1:printf("enter the item \n");
             scanf("%d",&item);
             head=insert_rear(item,head);
             break;

  case 2:printf("enter the key item \n");
             scanf("%d",&item);
             head=insert_leftpos(item,head);
             break;

  case 3:printf("enter the position\n");
                         scanf("%d",&pos);
                         head=delete_position(pos,head);
                         break;
  case 4:display(head);
             break;
 default:exit(0);

 }
}

}
```



```
[nishchal@Nishchals-MacBook-Pro desktop % ./a.out

 1:Insert at rear
 2:insert to left of key item
 3:Delete at a position
 4:display the linked list
 5:Exit
enter the choice
1
enter the item
22

 1:Insert at rear
 2:insert to left of key item
 3:Delete at a position
 4:display the linked list
 5:Exit
enter the choice
1
enter the item
33

 1:Insert at rear
 2:insert to left of key item
 3:Delete at a position
 4:display the linked list
 5:Exit
enter the choice
2
enter the key item
34
key not found

 1:Insert at rear
 2:insert to left of key item
 3:Delete at a position
 4:display the linked list
 5:Exit
enter the choice
2
enter the key item
33
enter item towards left of 33=2

 1:Insert at rear
 2:insert to left of key item
 3:Delete at a position
 4:display the linked list
 5:Exit
enter the choice
4
contents of list :
22
2
33

 1:Insert at rear
 2:insert to left of key item
 3:Delete at a position
 4:display the linked list
 5:Exit
enter the choice
```

## Lab Program 10:

Write a program a) To construct a binary Search tree. b) To traverse the tree using all the methods i.e., in-order, preorder and post order c) To display the elements in the tree

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

NODE getnode(int item)
 {
   NODE x = (NODE)malloc(sizeof(struct node));
   if(x!=NULL){
        x->data=item;
        x->left = NULL;
        x->right = NULL;
        return x;
        }
   else {
      printf("Memory allocation failed!\n");
      exit(0);
   }
}
NODE insert(NODE root,int item)
{
        if(root ==NULL)
                return getnode(item);
        if(item<root->data)
                root->left = insert(root->left,item);
        else if(item>root->data)
                root->right = insert(root->right,item);
        return root;
}
void inorder(NODE root)
{
        if(root == NULL)
        return;
        inorder(root->left);
        printf("%d\t",root->data);
        inorder(root->right);
}
void preorder(NODE root)
{
        if(root == NULL)
        return;
        printf("%d\t",root->data);
        preorder(root->left);
        preorder(root->right);
}
void postorder(NODE root)
{
        if(root == NULL)
        return;
        postorder(root->left);
        postorder(root->right);
        printf("%d\t",root->data);
}
int main()
{
        NODE root = NULL;
        int item,ch;
        for(;;)
        {
        printf("1.Insert.\n2.Inorder Traversal.\n3.Preorder Traversal.\n4.Postorder Traversal.\n5.Exit:
\n");
        scanf("%d",&ch);
        switch(ch){
                case 1: printf("\nEnter the element:\n");
                        scanf("%d",&item);
                        root = insert(root,item);
                        break;
                case 2: inorder(root);
                        break;
                case 3: preorder(root);
                        break;
                case 4: postorder(root);
                        break;
                case 5: exit(1);
                default :printf("Invalid Choice");
                }
        }
}
```