Lab session -4

Q1:

```
#include<stdio.h>
#include<stdlib.h>
struct node{
int info;
struct node * link;
}* front =NULL ,* rear =NULL;
void insert(int item);
int del();
int peek();
int isEmpty();
void display();
int main() {
int choice,item;
while(1){
printf("1.Insert\n");
printf("2.Delete\n");
printf("3.Display the element at the front \n");
printf("4.Display all elements of the queue \n");
printf("5.Quit\n\n");
printf("Enter your choice : ");
scanf("%d" ,&choice);
switch(choice){
case 1:
printf("Enter the element for adding in queue: ");
scanf("%d",&item);
insert(item);
break;
case 2 :
item=del();
printf("Deleted item is : %d\n ",item);
break;
case 3 :
printf("Item at the front of queue is : %d\n ",peek());
break;
case 4 :
display();
break;
case 5 :
exit(1);
default :
printf("Wrong choice\n");
}
}
return 0;
}
void insert(int item) {
struct node * tmp;
tmp=(struct node*) malloc(sizeof(struct node));
if( tmp == NULL){
printf( "Memory not available" );
return ;
}
tmp->info=item;
tmp->link=NULL;
if( front == NULL)
front=tmp;
else
```

```
rear->link=tmp;
rear=tmp;
}

int peek(){
if(isEmpty()){
printf("Queue Underflow\n");
exit(1);
}
return front->info;
}
int del() {
struct node * tmp;
int item;
if( isEmpty()){
printf( "Queue Underflow\n" );
exit(1) ;
}
tmp=front;
item = tmp->info;
front= front->link;
free(tmp);
return item;
}
int isEmpty() {
if( front==NULL)
return 1;
else
return 0;
}

void display (){
struct node *ptr;
ptr=front;
if(isEmpty()){
printf("Queue is empty\n");
return ;
}
printf("Queue elements :\n\n");
while(ptr!=NULL) {
printf("%d \n", ptr->info);
ptr = ptr->link;
}
printf("\n");
}
```

OUTPUT:

```
1.Insert
2.Delete
3.Display the element at the front
4.Display all elements of the queue
5.Quit

Enter your choice : 1
Enter the element for adding in queue: 1
1.Insert
2.Delete
3.Display the element at the front
4.Display all elements of the queue
5.Quit

Enter your choice : 1
Enter the element for adding in queue: 2
```

1.Insert

2.Delete

3.Display the element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 1

Enter the element for adding in queue: 3

1.Insert

2.Delete

3.Display the element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 1

Enter the element for adding in queue: 4

1.Insert

2.Delete

3.Display the element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 1

Enter the element for adding in queue: 5

1.Insert

2.Delete

3.Display the element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 2

Deleted item is : 1

1.Insert

2.Delete

3.Display the element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 3

Item at the front of queue is : 2

1.Insert

2.Delete

3.Display the element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 4

Queue elements :

2

3

4

5

1.Insert

2.Delete

3.Display the element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 5

Q2:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
struct node
{
   char info;
   struct node *link;
} *top = NULL;
char postfix[30];
char cop[30];
void push(char item);
char pop();
char peek();
int isEmpty();
void display();
void copy();
int isoperator(char item);
int main()
{
   int choice;
   char item;
   int a, b, temp;
   int result;
   char ele;
   char symbol;
   while (1)
   {
      printf("1.Push\n");
      printf("2.Pop\n");
      printf("3.Display the top element\n");
      printf("4.Display all the stack
elements\n");
      printf("5.Postfix to infix form\n");
      printf("6.Quit\n\n");
      printf("Enter your choice : ");
      scanf("%d", &choice);
      switch (choice)
      {
      case 1:
         printf("Enter the item to be pushed: ");
         getchar();
         scanf("%c", &item);
         push(item);
         break;
      case 2:
         item = pop();
         printf("Popped item is : %c\n ", item);
         break;
      case 3:
         printf("Item at the top is : %c\n ",
peek());
         break;
      case 4:
         display();
         break;
      case 5:
         printf("\nEnter the expression in postfix
form : ");
         scanf("%s", postfix);
         int i = 0;
```

```
        while (i < strlen(postfix))
        {
            ele = postfix[i];
            if (isoperator(ele) == 0)
            {
                push(ele);
                push(' ');
            }
            if (isoperator(ele) == 1)
            {
                struct node *p = top->link;
                while (p != NULL)
                {
                    if (p->info == ' ')
                    {
                        p->info = ele;
                        break;
                    }
                    p = p->link;
                }
            }
            i++;
        }
        copy();
        break;
    case 6:
        exit(1);
    default:
        printf("Wrong choice\n");
    }
}
    return 0;
}
void push(char item)
{
    struct node *tmp;
    tmp = (struct node *)malloc(sizeof(struct
                                node));
    if (tmp == NULL)
    {
        printf("Stack Overflow\n");
        return;
    }
    tmp->info = item;
    tmp->link = top;
    top = tmp;
}
char pop()
{
    struct node *tmp;
    char item;
    if (isEmpty())
    {
        printf("Stack Underflow\n");
        exit(1);
    }
    tmp = top;
    item = tmp->info;
    top = top->link;
    free(tmp);
    return item;
}
```

```
void display()
{
   struct node *ptr;
   ptr = top;
   if (isEmpty())
   {
      printf("Stack is empty\n");
      return;
   }
   printf("Stack elements :\n\n");
   while (ptr != NULL)
   {
      printf("%c ", ptr->info);
      ptr = ptr->link;
   }
   printf("\n");
}
char peek()
{
   if (isEmpty())
   {
      printf("Stack Underflow\n");
      exit(1);
   }
   return top->info;
}
int isEmpty()
{
   if (top == NULL)
      return 1;
   else
      return 0;
}
int isoperator(char ele)
{
   if ((ele >= 'A' && ele <= 'Z') || (ele >= 'a' &&
ele <= 'z'))
   {
      return 0;
   }
   else
   {
      return 1;
   }
}
void copy()
{
   struct node *p = top;
   int i = 0;
   while (p != NULL)
   {
      cop[i] = p->info;
      p = p->link;
      i++;
   }
   int l = strlen(cop) - 1;
   while (l >= 0)
   {
      printf("%c ", cop[l]);
      l--;}
printf("\n");
}
```

OUTPUT:

1.Push

2.Pop

3.Display the top element

4.Display all the stack elements

5.Postfix to infix form

6.Quit

Enter your choice : 1

Enter the item to be pushed: 1

1.Push

2.Pop

3.Display the top element

4.Display all the stack elements

5.Postfix to infix form

6.Quit

Enter your choice : 1

Enter the item to be pushed: 2

1.Push

2.Pop

3.Display the top element

4.Display all the stack elements

5.Postfix to infix form

6.Quit

Enter your choice : 4

Stack elements :

2 1

1.Push

2.Pop

3.Display the top element

4.Display all the stack elements

5.Postfix to infix form

6.Quit

Enter your choice : 2

Popped item is : 2

1.Push

2.Pop

3.Display the top element

4.Display all the stack elements

5.Postfix to infix form

6.Quit

Enter your choice : 2

Popped item is : 1

1.Push

2.Pop

3.Display the top element

4.Display all the stack elements

5.Postfix to infix form

6.Quit

Enter your choice : 5

Enter the expression in postfix form : abc/+*

a + b / c

1.Push

2.Pop

3.Display the top element

4.Display all the stack elements

5.Postfix to infix form

6.Quit

Enter your choice : 6