SANGHAMITRA R
LAB 2 - Queues and postfix expression

1. Write a program 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), / (divide) and ^ (power).

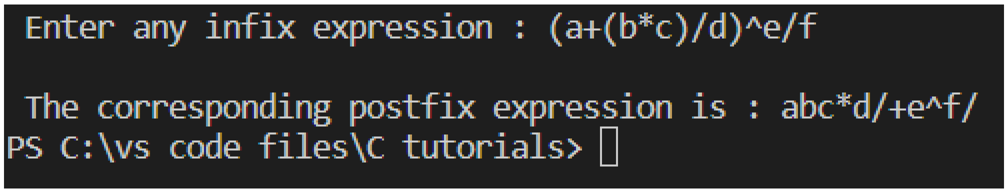
```c
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>
#define MAX 100
char st[MAX];
int top =-1;
void push(char st[], char);
char pop(char st[]);
void InfixtoPostfix(char source[], char target[]);
int getpri(char);
int main()
{
char infix[100], postfix[100];
printf("\n Enter any infix expression : ");
scanf("%s",infix);
strcpy(postfix, "");
InfixtoPostfix(infix, postfix);
printf("\n The corresponding postfix expression is : ");
puts(postfix);
}
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] == ')')
```

```c
 {
 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] == '%' || source[i] == '^')
 {
 while ((top !=-1) && (st[top] != '(') && (getpri(st[top]) >
getpri(source[i])))
 {
 target[j] = pop(st);
 j++;
 }
 push(st, source[i]);
 i++;
 }
 else
 {
 printf("\n INCORRECT ELEMENT IN EXPRESSION");
 exit(1);
 }
 }
 while ((top !=-1) && (st[top] != '('))
 {
```

```
 target[j] = pop(st);
 j++;
 }
 target[j] = '\0';
 }
 int getpri(char op)
{
 if (op == '^')
 return 2;
 else 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 OVERFLOW");
 else
 {
 }
 top++;
 st[top] = val;
 }
 char pop(char st[])
 {
 char val = ' ';
 if (top ==-1)
 printf("\n STACK UNDERFLOW");
 else
 {
 val = st[top];
 top--;
 }
 return val;
 }
```

OUTPUT:

```
Enter any infix expression : (a+(b*c)/d)^e/f
The corresponding postfix expression is : abc*d/+e^f/
```



```
 Enter any infix expression : (a+(b*c)/d)^e/f

 The corresponding postfix expression is : abc*d/+e^f/
PS C:\vs code files\C tutorials> 
```

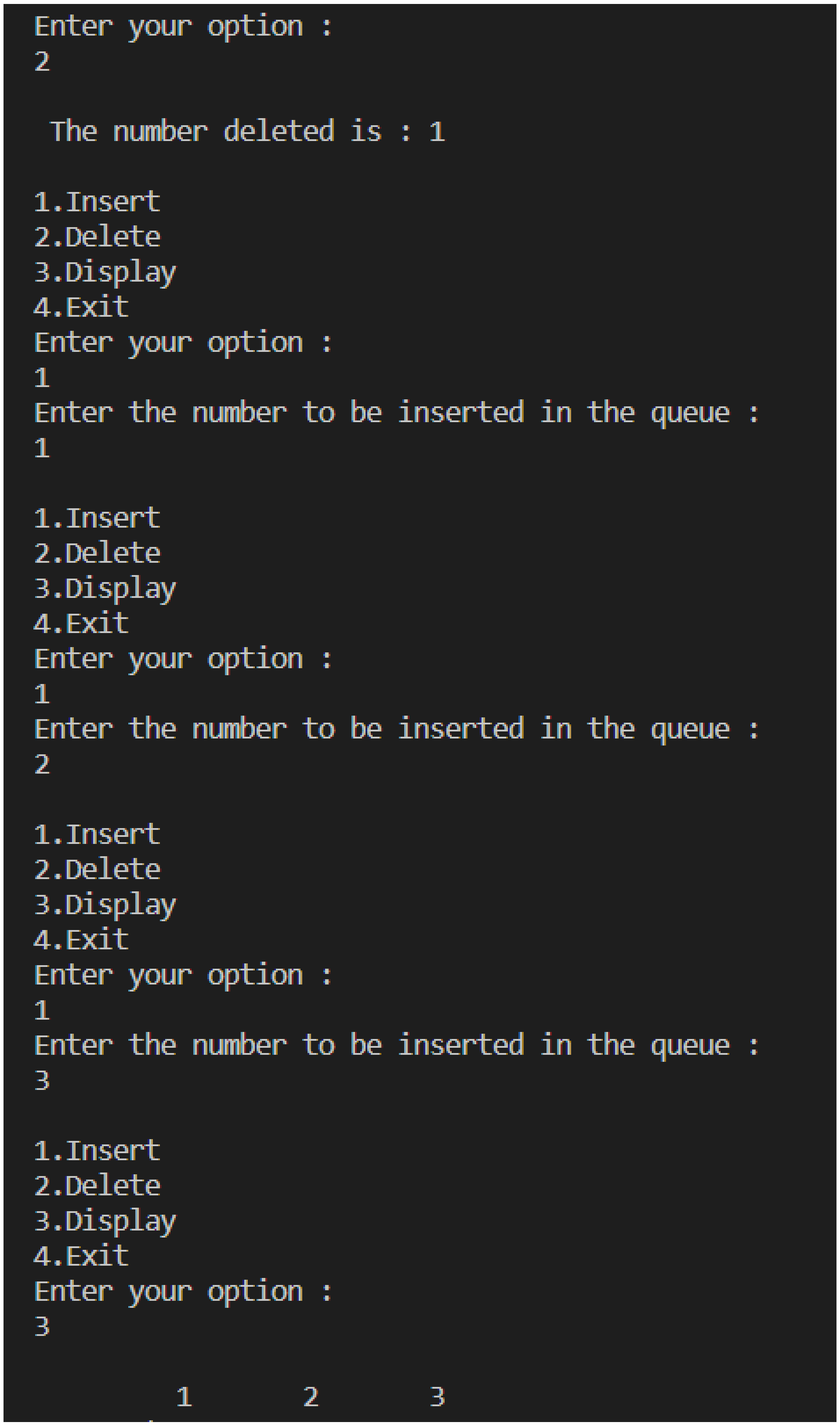
 2. WAP to simulate the working of a queue of integers using an array. Provide the following operations
a) Insert
b) Delete
c) Display
The program should print appropriate messages for queue empty and queue overflow conditions.

```
#include <stdio.h>
#include <stdlib.h>
#define N 5
int q[N];
int front =-1, rear =-1;
void insert(int);
int deleteq();
void display();
int main()
{
int n, choice;
do
{
printf("\n1.Insert\n2.Delete\n3.Display\n4.Exit\n");
printf("Enter your option : \n");
scanf("%d", &choice);
switch (choice)
{
case 1:
printf("Enter the number to be inserted in the queue : \n");
scanf("%d", &n);
insert(n);
break;
case 2:
n =deleteq();
if (n !=-1)
```

```
 printf("\n The number deleted is : %d\n", n);
 break;
 case 3:
 display();
 break;
 case 4:
 exit(0);
 break;
 default:
 printf("Invalid option\n");
 exit(0);
 break;
 }
 } while (choice != 4);
}
 void insert(int num)
 {
 if (rear == N- 1)
 printf("\n OVERFLOW");
 else if (front ==-1 && rear ==-1)
 front = rear = 0;
 else
 rear++;
 q[rear] = num;
 }
 int deleteq()
 {
 int val;
 if (front ==-1 || front > rear)
 {
 printf("\n UNDERFLOW");
 return-1;
 }
 else
 {
 val = q[front];
 front++;
 if (front > rear)
 front = rear =-1;
 return val;
```

```
}
}
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", q[i]);
}
}
```

OUTPUT



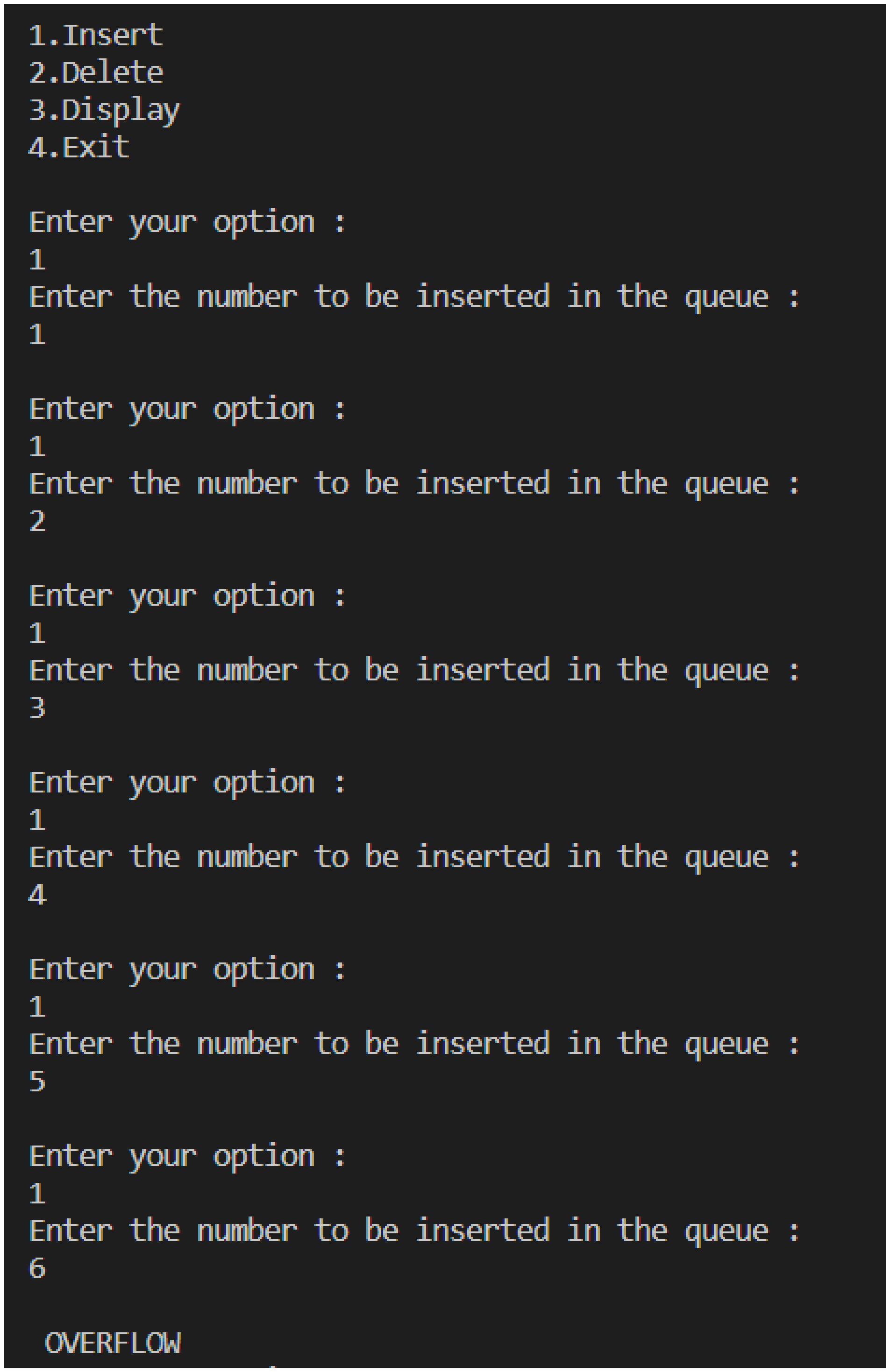
```
Enter your option :
2

 The number deleted is : 1

1.Insert
2.Delete
3.Display
4.Exit
Enter your option :
1
Enter the number to be inserted in the queue :
1

1.Insert
2.Delete
3.Display
4.Exit
Enter your option :
1
Enter the number to be inserted in the queue :
2

1.Insert
2.Delete
3.Display
4.Exit
Enter your option :
1
Enter the number to be inserted in the queue :
3

1.Insert
2.Delete
3.Display
4.Exit
Enter your option :
3

         1       2       3
```

3. 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.

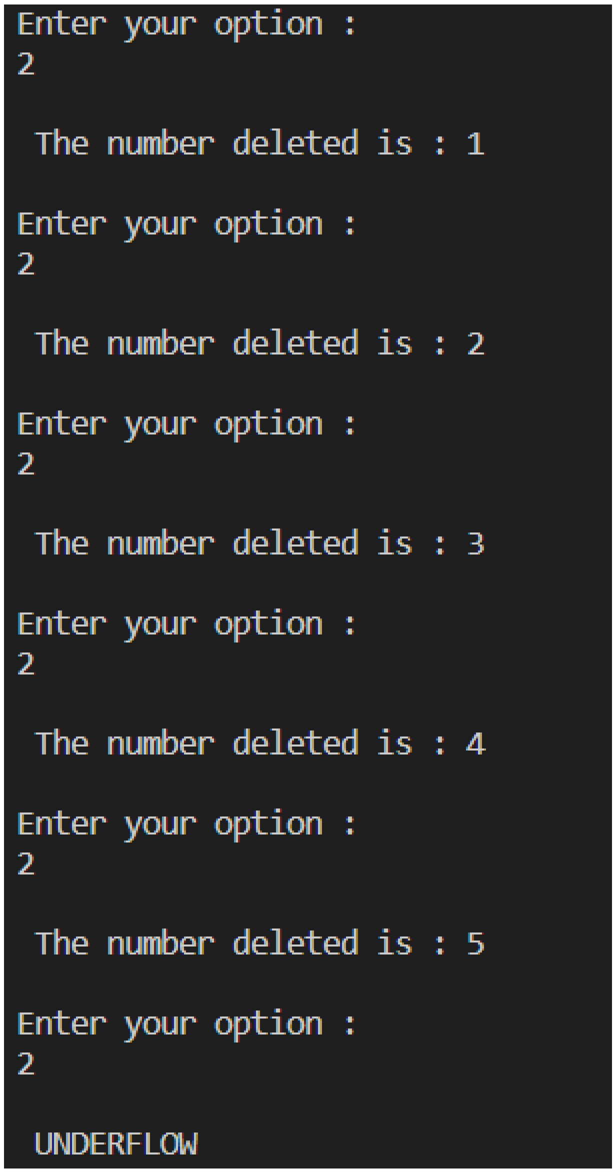
```c
#include <stdio.h>
#include <stdlib.h>
#define N 5
int q[N];
int front = -1, rear = -1;
void insert(int);
int deleteq();
void display();
int main()
{
int n, choice;
printf("\n1.Insert\n2.Delete\n3.Display\n4.Exit\n");
do
{
printf("\nEnter your option : \n");
scanf("%d", &choice);
switch (choice)
{
case 1:
printf("Enter the number to be inserted in the queue : \n");
scanf("%d", &n);
insert(n);
break;
case 2:
n = deleteq();
if (n != -1)
printf("\n The number deleted is : %d\n", n);
break;
case 3:
display();
break;
case 4:
exit(0);
break;
default:
printf("Invalid option\n");
exit(0);
break;
}
} while (choice != 4);
}
void insert(int num)
{
if ((front == 0 && rear == N - 1) || (rear == (front - 1)))
printf("\n OVERFLOW");
else if (front == -1 && rear == -1)
{
front = rear = 0;
q[rear] = num;
}
else if (rear == N - 1 && front != 0)
{
rear = 0;
q[rear] = num;
}
else
{
rear++;
q[rear] = num;
}
}
int deleteq()
{
int val;
if (front == -1 && rear == -1)
{
printf("\n UNDERFLOW");
return -1;
}
val = q[front];
if (front == rear)
front = rear = -1;
else
```

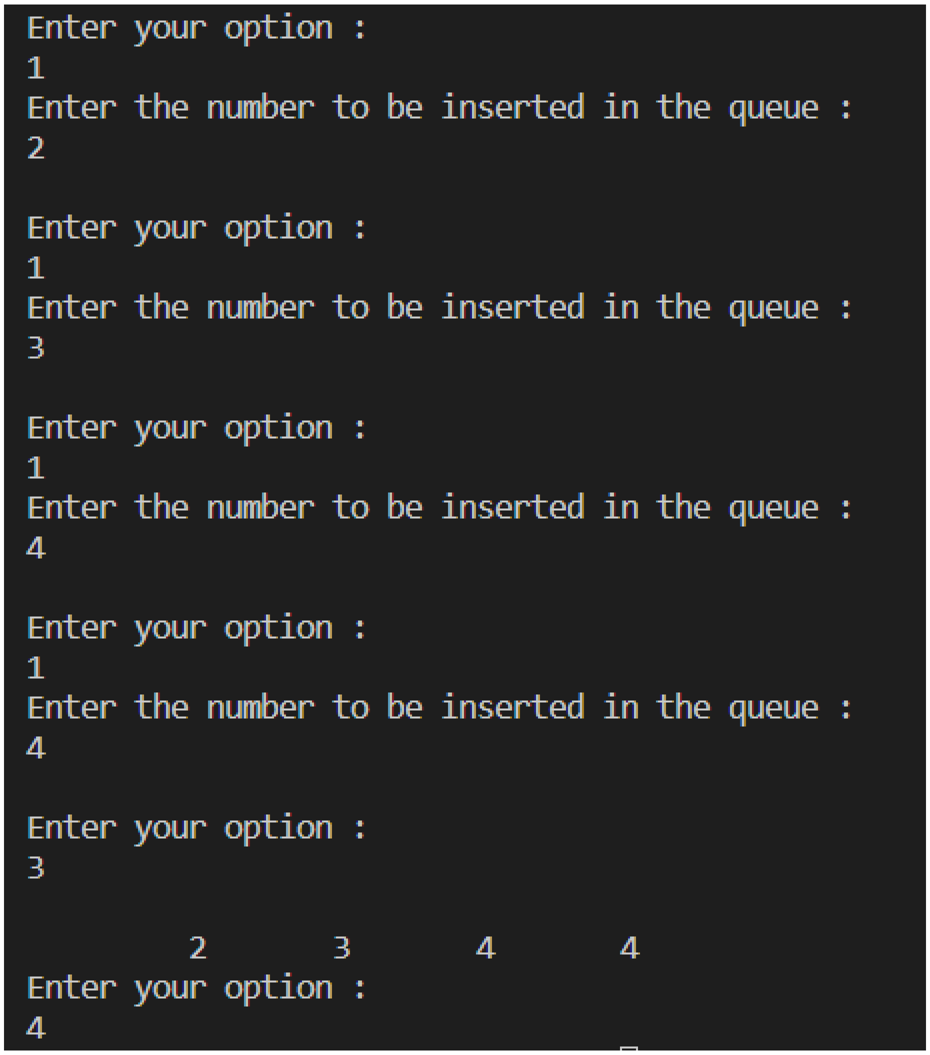
```
{
if (front == N - 1)
front = 0;
else
front++;
}
return val;
}
void display()
{
int i;
printf("\n");
if (front == -1 && rear == -1)
printf("\n QUEUE IS EMPTY");
else
{
if (front < rear)
{
for (i = front; i <= rear; i++)
printf("\t %d", q[i]);
}
else
{
for (i = front; i < N; i++)
printf("\t %d", q[i]);
for (i = 0; i <= rear; i++)
printf("\t %d", q[i]);
}
}
}
```

OUTPUT:



```
1.Insert
2.Delete
3.Display
4.Exit

Enter your option :
1
Enter the number to be inserted in the queue :
1

Enter your option :
1
Enter the number to be inserted in the queue :
2

Enter your option :
1
Enter the number to be inserted in the queue :
3

Enter your option :
1
Enter the number to be inserted in the queue :
4

Enter your option :
1
Enter the number to be inserted in the queue :
5

Enter your option :
1
Enter the number to be inserted in the queue :
6

 OVERFLOW
```


```
Enter your option :
2

 The number deleted is : 1

Enter your option :
2

 The number deleted is : 2

Enter your option :
2

 The number deleted is : 3

Enter your option :
2

 The number deleted is : 4

Enter your option :
2

 The number deleted is : 5

Enter your option :
2

 UNDERFLOW
```



```
Enter your option :
1
Enter the number to be inserted in the queue :
2

Enter your option :
1
Enter the number to be inserted in the queue :
3

Enter your option :
1
Enter the number to be inserted in the queue :
4

Enter your option :
1
Enter the number to be inserted in the queue :
4

Enter your option :
3

         2       3       4       4
Enter your option :
4
```