LAB-3 01/01/2024

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).

Source Code:

```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#define MAX 100
char st[MAX];
int top = -1;
void infixtopostfix(char source[], char target[]);
int getpriority(char);
void push(char st[], char);
char pop(char st[]);
int main()
{
    char infix[100], postfix[100];
    printf("Enter any infix expression\n");
    gets(infix);
    strcpy(postfix, "");
    infixtopostfix(infix, postfix);
    printf("The corresponding postfix expression is:\n");
    puts(postfix);
```

```
    return 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("Stack overflow\n");
    else
    {
        top++;
        st[top] = val;
    }
}
char pop(char st[])
{
    char val = ' ';
    if (top == -1)
    {
        printf("Stack Underflow\n");
```

```
        }
        else
        {
                val = st[top];
                top--;
        }
        return val;
}

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 Element in Expression ");
                exit(1);
            }
        }
        while ((top != -1) && (st[top] != '('))
        {
            target[j] = pop(st);
            j++;
        }
        target[j] = '\0';
}
```

OUTPUT:

Enter any infix expression

A+BC*

The corresponding postfix expression is:

ABC*+

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.

Source Code:

```
#include <stdio.h>

#include <stdlib.h>


#define MAX 2

int q[MAX];

int front = -1;

int rear = -1;


void insert();

int delete();

void display();


int main()

{

    while (1)

    {
```

```
        Int n,choice;
        printf("\n1. insert \t 2.delete \t 3.display \t 4.exit\n");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            insert();
            break;
        case 2:
            n=delete();
             printf(“%d is deleted from queue”,n);
            break;
        case 3:
            display();
            break;
        case 4:
            exit(0);
        }
    }
}

void insert()
{
    if (rear == MAX - 1)
    {
```

```
            printf("Queue is Full\n");
            return;
        }
        printf("Enter the element to be inserted\n");
        int a;
        scanf("%d", &a);
        if ((front == -1) && (rear == -1))
        {
            front = rear = 0;
        }
        else
        {
            rear++;
        }
        q[rear] = a;
}

int delete()
{
        if (front == rear && rear != 0)
        {
            printf("queue is empty");
            exit(0);
        }
        if (front == rear == 0)
```

```
    {
        int x = q[front];
        front = rear = -1;
        return x;
    }
    else if (front != -1 && rear > front)
    {
        int x = q[front];
        front++;
        return x;
    }
}
void display()
{
    printf("the elements are:\n");
    for (int i = front; i <= rear; i++)
    {
        printf("%d \n", q[i]);
    }
}
```

OUTPUT:

```
1. insert        2.delete        3.display        4.exit
1
Enter the element to be inserted
12
```

1. insert 2.delete 3.display 4.exit

1

Enter the element to be inserted

13

1. insert 2.delete 3.display 4.exit

1

Queue is Full

1. insert 2.delete 3.display 4.exit

3

The elements are :

12

13

1. insert 2.delete 3.display 4.exit

2

12 is deleted from the queue

1. insert 2.delete 3.display 4.exit

2

13 is deleted from the queue

1. insert 2.delete 3.display 4.exit

2

Queue is empty

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.

Source Code:

```
#include <stdio.h>

#define MAX 10

int queue[MAX];

int front = -1, rear = -1;

void insert(void);

int delete(void);

int peek(void);

void display(void);


void main()

{

    int option, val;

    do

    {

        printf("Enter 1.Insert 2.Delete 3.Peek 4.Display 5.Exit : ");
```

```
scanf("%d", &option);
switch (option)
{
case 1:
        insert();
        break;

case 2:
        val = delete ();
        if (val != -1)
                printf("the number deleted is : %d", val);
        break;

case 3:
        val = peek();
        if (val != -1)
                printf("the first value in queue is : %d", val);
        break;

case 4:
        display();
        break;

case 5:
        return;
```

```
            default:
                printf("Incorrect exp");
            }
        } while (option != 5);
}
void insert()
{
        int num;
        printf("Enter the numbeer : ");
        scanf("%d", &num);

        if (front == 0 && rear == MAX - 1)
            printf("Overflow");
        else if (front == -1 && rear == -1)
        {
            front = rear = 0;
            queue[rear] = num;
        }
        else if (rear == MAX - 1 && front != 0)
        {
            rear = (++rear) % MAX;
            queue[rear] = num;
        }
        else
```

```
    {
        rear++;
        queue[rear] = num;
    }
}
int delete()
{
    int val;
    if (front == -1 && rear == -1)
    {
        printf("Underflow");
        return -1;
    }
    val = queue[front];
    if (front == rear)
    {
        front = rear = -1;
    }
    else
    {
        if (front == MAX - 1)
            front = 0;
        else
            front++;
    }
```

```
        return val;
}
int peek()
{
        if (front == -1 && rear == -1)
        {
                printf("Queue is empty");
                return -1;
        }
        else
        {
                return queue[front];
        }
}
void display()
{
        int i;
        if (front == -1 && rear == -1)
                printf("Queue is empty");
        else
        {
                if (front < rear)
                {
                        for (i = front; i <= rear; i++)
                        {
```

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

OUTPUT:

Enter 1.Insert 2.Delete 3.Peek 4.Display 5.Exit :

1

Enter the number : 10

Enter 1.Insert 2.Delete 3.Peek 4.Display 5.Exit :

1

Enter the number : 20

Enter 1.Insert 2.Delete 3.Peek 4.Display 5.Exit :

2

the number deleted is : 10

Enter 1.Insert 2.Delete 3.Peek 4.Display 5.Exit :

1

Enter the number : 30

Enter 1.Insert 2.Delete 3.Peek 4.Display 5.Exit :

4

20 30

Enter 1.Insert 2.Delete 3.Peek 4.Display 5.Exit :

3

the first value in queue is : 20

Enter 1.Insert 2.Delete 3.Peek 4.Display 5.Exit :

3

the first value in queue is : 20

Enter 1.Insert 2.Delete 3.Peek 4.Display 5.Exit :

1

Enter the number : 40

Enter 1.Insert 2.Delete 3.Peek 4.Display 5.Exit :

4

20 30 40

Enter 1.Insert 2.Delete 3.Peek 4.Display 5.Exit :

2

the number deleted is : 20

Enter 1.Insert 2.Delete 3.Peek 4.Display 5.Exit :

5