**Lab program 2**

**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) and / (divide)**

**#include <stdio.h>**

**#include <ctype.h>**

**#include <string.h>**

**#define MAX 100**

**char stack[MAX];**

**int top = -1;**

**void push(char c) {**

**if (top == MAX - 1) {**

**printf("Stack overflow\n");**

**} else {**

**stack[++top] = c;**

**}**

**}**

**char pop() {**

**if (top == -1) {**

**printf("Stack underflow\n");**

**return -1;**

**} else {**

**return stack[top--];**

**}**

**}**

**int precedence(char c) {**

**if (c == '+' || c == '-') {**

**return 1;**

**} else if (c == '\*' || c == '/') {**

**return 2;**

**}**

**return 0;**

**}**

**void infixToPostfix(char\* expression) {**

**char postfix[MAX];**

**int i, j = 0;**

**for (i = 0; expression[i] != '\0'; i++) {**

**char ch = expression[i];**

**if (isalnum(ch)) {**

**postfix[j++] = ch;**

**}**

**else if (ch == '(') {**

**push(ch);**

**}**

**else if (ch == ')') {**

**while (top != -1 && stack[top] != '(') {**

**postfix[j++] = pop();**

**}**

**pop(); // Remove '(' from the stack**

**}**

**else {**

**while (top != -1 && precedence(stack[top]) >= precedence(ch)) {**

**postfix[j++] = pop();**

**}**

**push(ch);**

**}**

**}**

**while (top != -1) {**

**postfix[j++] = pop();**

**}**

**postfix[j] = '\0';**

**printf("Postfix expression: %s\n", postfix);**

**}**

**int main() {**

**char expression[MAX];**

**printf("Enter a valid infix expression: ");**

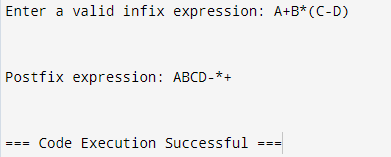
**scanf("%s", expression);**

**infixToPostfix(expression);**

**return 0;**

**}**

**Output:**

****

**Lab program 3**

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

**#include <stdio.h>**

**#define MAX 5**

**int queue[MAX];**

**int front = -1, rear = -1;**

**void insert(int value) {**

**if (rear == MAX - 1) {**

**printf("Queue Overflow! Cannot insert %d\n", value);**

**} else {**

**if (front == -1) {**

**front = 0; // Set front to 0 when the first element is inserted**

**}**

**rear++;**

**queue[rear] = value;**

**printf("%d inserted into the queue\n", value);**

**}**

**}**

**void delete() {**

**if (front == -1 || front > rear) {**

**printf("Queue Underflow! Cannot delete from an empty queue\n");**

**} else {**

**printf("%d deleted from the queue\n", queue[front]);**

**front++;**

**if (front > rear) {**

**front = rear = -1;**

**}**

**}**

**}**

**void display() {**

**if (front == -1) {**

**// Queue empty condition**

**printf("The queue is empty\n");**

**} else {**

**printf("Queue elements: ");**

**for (int i = front; i <= rear; i++) {**

**printf("%d ", queue[i]);**

**}**

**printf("\n");**

**}**

**}**

**int main() {**

**int choice, value;**

**do {**

**printf("\nQueue Operations:\n");**

**printf("1. Insert (Enqueue)\n");**

**printf("2. Delete (Dequeue)\n");**

**printf("3. Display\n");**

**printf("4. Exit\n");**

**printf("Enter your choice: ");**

**scanf("%d", &choice);**

**switch (choice) {**

**case 1:**

**printf("Enter value to insert: ");**

**scanf("%d", &value);**

**insert(value); // Call insert function**

**break;**

**case 2:**

**delete(); // Call delete function**

**break;**

**case 3:**

**display(); // Call display function**

**break;**

**case 4:**

**printf("Exiting...\n");**

**break;**

**default:**

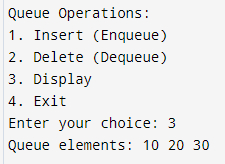
**printf("Invalid choice! Please try again.\n");**

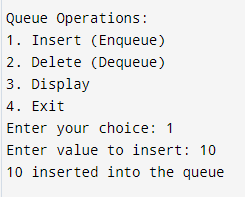
**}**

**} while (choice != 4);**

**return 0;**

**}**

**Output: **

****

**WAP to simulate the working of a circular queue of**

**integers using an array. Provide the following operations:**

**Insert, Delete & Display**

**The program should print appropriate messages for queue**

**empty and queue overflow conditions**

**#include <stdio.h>**

**#define MAX 3**

**int queue[MAX];**

**int front = -1, rear = -1;**

**void insert(int value) {**

**if ((front == 0 && rear == MAX - 1) || (rear == (front - 1) % (MAX - 1))) {**

**printf("Queue Overflow\n");**

**return;**

**} else if (front == -1) {**

**front = rear = 0;**

**queue[rear] = value;**

**} else if (rear == MAX - 1 && front != 0) {**

**rear = 0;**

**queue[rear] = value;**

**} else {**

**rear++;**

**queue[rear] = value;**

**}**

**printf("Inserted element: %d\n", value);**

**}**

**void delete() {**

**if (front == -1) {**

**printf("Queue Underflow\n");**

**return;**

**}**

**printf("Deleted element: %d\n", queue[front]);**

**if (front == rear) {**

**front = rear = -1;**

**} else if (front == MAX - 1) {**

**front = 0;**

**} else {**

**front++;**

**}**

**}**

**void display() {**

**if (front == -1) {**

**printf("Queue is Empty\n");**

**return;**

**}**

**printf("Queue elements are: ");**

**if (rear >= front) {**

**for (int i = front; i <= rear; i++)**

**printf("%d ", queue[i]);**

**} else { // Circular condition**

**for (int i = front; i < MAX; i++)**

**printf("%d ", queue[i]);**

**for (int i = 0; i <= rear; i++)**

**printf("%d ", queue[i]);**

**}**

**printf("\n");**

**}**

**int main() {**

**int choice, value;**

**while (1) {**

**printf("\nCircular Queue Operations:\n");**

**printf("1. Insert\n");**

**printf("2. Delete\n");**

**printf("3. Display\n");**

**printf("4. Exit\n");**

**printf("Enter your choice: ");**

**scanf("%d", &choice);**

**switch (choice) {**

**case 1:**

**printf("Enter the value to insert: ");**

**scanf("%d", &value);**

**insert(value);**

**break;**

**case 2:**

**delete();**

**break;**

**case 3:**

**display();**

**break;**

**case 4:**

**printf("Exiting...\n");**

**return 0;**

**default:**

**printf("Invalid choice! Please try again.\n");**

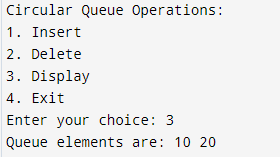
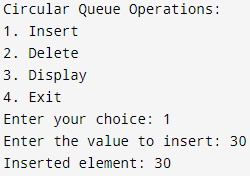
**}**

**}**

**return 0;**

**}**

**Output:**

** **