01/01/24

Q. Lab-2

WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 100

char stack[MAX];
char infix[MAX], postfix[MAX];
int top = -1;

void push(char);
char pop();
int isEmpty();
void inToPost();
void print();
int precedence(char);

int main()
{
printf("Enter the infix expression: ");
gets(infix);
inToPost();
print();
return 0;
}

void inToPost()
{
int i, j=0;
```

```
char symbol, next;
for(i=0; i<strlen(infix); i++)
  {
    symbol = infix[i];
switch(symbol)
  {
    case ')':
      while((next = pop()) != '(')
        postfix[j++] = next;
      break;
    case '+':
    case '-':
    case '*':
    case '/':
    case '^':
while(!isEmpty() && precedence(stack[top]) >=
            precedence(symbol))
    {
      push(symbol);
      break;
default:
      postfix[j++] = symbol;
  }
}
  while(!isEmpty())
      postfix[j++] = pop();
        postfix[j] = '\0';
}

void print()
  {
    int i=0;
  printf("\n The equivalent postfix expression: \n");
    while(postfix[i])
```

```
    {
    printf("%c", postfix[i+1]);
    }
    printf("\n");
}
void push(char c)
{
if (top == MAX-1)
  {
  printf("Stack Overflow\n");
  return;
}
  top++;
  stack[top] = c;
}
char pop()
  {
  char c;
  if (top == -1)
    {
    printf("Stack underflow\n");
    exit(1);
  }
  return(stack[top--]);
}

int isEmpty()
  {
  if (top == -1)
    return 1;
  else
    return 0;
}
```

```
int precedence (char symbol)
{
  switch (symbol)
  {
    case '^':
      return 3;
    case '/':
    case '*':
      return 2;
    case '#':
    case '-':
      return 1;
    default:
      return 0;
  }
}
```

Op: Enter the infix exp : A+B-C*D
Postfix expression: AB+CD*-

Q) C00P to simulate the working of a queue of integers using an array. Provide insert, delete, display

```
#include <stdio.h>
#include <conio.h>
#define MAX 10
int queue[MAX];
int front = -1, rear = -1;
void insert (void);
int delete_element (void);
int peek (void);
void display (void);
int main()
{
  int option, val;
  do
  {
```

```
printf("\n1. INSERT AN ELEMENT");
printf("\n2. Delete an Element");
printf("\n3. Display the queue");
printf("\n4. Exit");
printf("\n Enter your option");
scanf("%d", &option);
switch(option)
 {
case 1: insert();
        break;
case 2: delete();
        break;
case 3: display();
        break;
default : printf("Invalid choice\n");
  }
}
}

void display()
 {
   int i;
   if (front == -1)
      printf("Queue is empty\n");

   else
    {
     printf("Queue is:\n");
     for(i=front; i<=rear; i++)
        printf("%d", queue[i]);
    }
}
```

```
void insert ()
{
  int num;
  if (rear == MAX-1)
    printf ("Queue overflow \n");
  else
  {
    if (front == -1)
      front = 0;
    printf ("Insert the element in queue ");
    scanf ("%d", &num);
    rear = rear + 1;
    queue[rear] = num;
  }
}

void delete ()
{
  if (front == -1 || front > rear)
  {
    printf ("Queue underflow \n");
    return;
  }
  else
  {
    printf ("Deleted element is %d \n",
            queue[front]);
    front = front + 1;
  }
}
```

Output:

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

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

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

Q3 WAP to simulate the working of a circular queue:

```
#include<stdio.h>
# define max 8
int queue[max];
  int front=-1;
  int rear=-1;
```

```
void enqueue(int element)
{
  if (front == -1 && rear == -1)
  {
    front = 0;
    rear = 0;
    queue[rear] = element;
  }
  else if ((rear+1) % max == front)
  {
    printf("Queue is overflow..");
  }
  else
  {
    rear = (rear+1) % max;
    queue[rear] = element;
  }
}
int dequeue()
{
  if ((front == -1) && (rear == -1))
  {
    printf("\n Queue is underflow");
  }
  else if (front == rear)
  {
    printf("\n The dequeued element is %d",
        queue[front]);
    front = -1;
    rear = -1;
  }
  else
  {
    printf("\n The dequeued element is %d", queue[front]);
```

```
        front = (front + 1) % max;
    }
}

void display()
{
    int i = front;
    if (front == -1 && rear == -1)
    {
        printf("\n Queue is empty..");
    }
    else
    {
        printf("\n Elements in a Queue are :");
        while (i <= rear)
        {
            printf("%d", queue[i]);
            i = (i+1) % max;
        }
    }
}

int main()
{
    int choice = 1, x;
    while (choice < 4 && choice != 0)
    {
        printf("\n 1. Insert an element");
        printf("\n 2. Delete an element");
        printf("\n 3. Display");
        printf("\n Enter your choice");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1: printf("Enter the element which is to
```

```
be inserted :");
    scanf("%d", &x);
    enqueue(x);
    break;
case 2: dequeue();
    break;
case 3: display();
  }
}
  return 0;
}
```

Output:

```
1. Insert
2. Display
3. Delete
Enter your choice 1
Enter the element to be inserted 5

1. Insert
2. Display
3. Delete
Enter your choice 1
Enter the element to be inserted 6

1. Insert
2. Display
3. Delete
Enter your choice 3
Elements in queue are 5,6.
```

R. 1/1/2024