URBAN EDGE 28/12/23

infix To Postfix

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

char stack[MAX];
int top = -1;
void push(char);
char pop();
int precedence(char);
void infixToPostfix(char infix[], char postfix[]);

void push(char item)
{
    if (top == MAX-1)
    {
        printf("Stack overflow.\n");
    }
    else
    {
        top++;
        stack[top] = item;
    }
}
char pop()
{
    if (top == -1)
    {
        printf("Stack underflow.\n");
    }
    else
    {
        char popped = stack[top];
        top--;
        return popped;
    }
}
```

```
int precedence (char Symbol)
{
    if (Symbol == '^')
    {
        return 3;
    }
    else if (Symbol == '*' || Symbol == '/')
        return 2;
    else if (Symbol == '+' || Symbol == '-')
        return 1;
    else
        return -1;
}
Void infixToPostfix (char infix[], char postfix[]);
{
    int i=0, j=0;
    char Symbol, temp;
    push ('#');

    while ((Symbol = infix [i++]) != '\0')
    {
        if (Symbol == '(')
        {
            push (Symbol);
        }
        else if (isalnum (Symbol))
        {
            postfix [j++] = Symbol;
        }
        else if (Symbol == ')')
        {
            while (Stack [top] != '(')
            {
                postfix [j++] = pop ();
            }
```

URBAN EDGE 28/12/23

```
            tem = pop ();
        }
        else
        {
            while (precedence (stack [top]) >= precedence
                (Symbol))
            {
                postfix [j++] = pop ();
            }
            push (Symbol);
        }
    }
    while (stack [top] != '#')
    {
        postfix [j++] = pop ();
    }
    postfix [j] = '\0';
}
int main()
{
    char infix [MAX], postfix [MAX];
    printf ("enter a valid infix expression with
        parenthesis: \n");
    scanf ("%s", infix);
    infixToPostfix (infix, postfix);
    printf ("the postfix expression is: \n %s \n", postfix);
    return 0;
}
```

output: enter a valid infix expression with parenthesis

AB*CD*+-E

the postfix expression is

ABCD**+E-

Postfix evaluation

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 100
// Stack implementation
int stack [MAX_SIZE];
int top = -1;
void push (int item)
{
    if (top >= MAX_SIZE - 1);
    {
        printf ("Stack overflow \n");
        return;
    }
    top ++;
    stack [top] = item;
}

int pop()
{
    if (top < 0)
    {
        printf ("Stack underflow \n");
        return -1;
    }
    int item = stack [top];
    top --;
    return item;
}
int is_operator (char symbol)
{
    if (symbol == '+' || symbol == '-' || symbol == '*'
        || symbol == '/')
    {
        return 1;
    }
    return 0;
}

int evaluate (char * expression)
{
    int i = 0;
    char symbol = expression [i];
    int operand1, operand2, result;

    while (symbol != '\0')
    {
        if (symbol >= '0' && symbol <= '9')
        {
            int num = symbol - '0';
            push (num);
        }
        else if (is_operator (symbol))
        {
            operand2 = pop();
            operand1 = pop();
            switch (symbol)
            {
                case '+': result = operand1 + operand2; break;
                case '-': result = operand1 - operand2; break;
                case '*': result = operand1 * operand2; break;
                case '/': result = operand1 / operand2; break;
            }
            push (result);
        }
        i ++;
        symbol = expression [i];
    }
    result = pop();
    return result;
}
```

```
int main()
{
    char expression [MAX_SIZE];
    printf ("enter the postfix expression : \n");
    gets ("%s", expression);
    int result = evaluate (expression);
    printf ("Result = %d \n", result);
    return 0;
}
```

output:

enter the postfix expression:

89*6+-*3*

Result = 632

Sol.)