28-12-23

Lab - 3

1) 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 ();
            temp = 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 infix expression: \n");
    scanf("%s", infix);
    infixToPostfix(infix, postfix);
    printf("The postfix expression is: %s \n", postfix);
    return 0;
}
```

Output:

Enter a infix expression:

a*b+c*d-e

The postfix expression is: ab*cd*+e-

2) Evaluating the postfix expression

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 100

int stack[MAX_SIZE];
int top = -1;
void push (int item)
{
    if (top >= MAX_SIZE -1)
        printf ("Stack overflow");
    top ++;
    stack[top] = item;
}

int pop ()
{
    if (top == -1)
        printf ("Stack underflow");
        return -1;
    int item = stak[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_operand (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 postfix expression \n");
    scanf("%s", expression);
    int result = evaluate(expression);
    printf("Result = %d \n", result);
    return 0;
}
```

Output:

Enter the postfix expression:

12 * 34 * + 5 -

result = 9