

Lab-2

Write a program to convert the given valid infix expression to postfix expression. The expression consist of single character operands and the binary operators +,-,* and / .

```
#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
    {
        top=top+1;
        stack[top]=c;
    }
}
char pop()
{
    char val;
    if(top== -1)
    {
        printf("stack underflow\n");
        return -1;
    }
    else
    {
        val=stack[top];
        top=top-1;
        return val;
    }
}

char peek()
{
    if(top== -1)
        return '\0';
    return stack[top];
}

int precedence(char c)
{
    if(c=='+'||c=='-')
        {return 1;}
}
```

```

        if(c=='*'||c=='/')
            {return 2;}
return 0;
}
void infixtopostfix(char infix[],char postfix[])
{
    int i,k=0;
    char c;
    for(i=0;infix[i]!='\0';i++)
    {
        c=infix[i];
        if(isalnum(c))
        {
            postfix[k]=c;
            k=k+1;
        }
        else if(c=='(')
        {
            push(c);
        }
        else if(c==')')
        {
            while(top!=-1&&peek()!='(')
            {
                postfix[k]=pop();
                k=k+1;
            }

            pop();
        }
        else
        {
            while(top!=-1&&precedence(peek())>=precedence(c))
            {
                postfix[k]=pop();
                k=k+1;
            }
            push(c);
        }
    }
    while(top!=-1)
    {
        postfix[k]=pop();
        k=k+1;
    }
    postfix[k]='\0';
}

```

```

int main()
{
    char infix[max],postfix[max];
    printf("enter the valid parenthesized infix expression:");
    scanf("%s",&infix);
    infixtopostfix(infix,postfix);
    printf("postfix expression:%s\n", postfix);
    return 0;
}

```

The image displays two screenshots of a Windows terminal window, showing the execution of a C program that converts infix expressions to postfix notation.

Top Screenshot:

```

C:\Users\Nijir\Documents> .\ida
enter the valid parenthesized infix expression:(a+b)/(c-d)-(e*f)
postfix expression:ab+cd-/ef*-
Process returned 0 (0x0)   execution time : 26.944 s
Press any key to continue.

```

Bottom Screenshot:

```

C:\Users\Nijir\Documents> .\ida
enter the valid parenthesized infix expression:((a+b-c+f*m)/n*k/(g+h))
postfix expression:ab+c-fm+n/k*gh+/
Process returned 0 (0x0)   execution time : 44.448 s
Press any key to continue.

```

The terminal window shows the program's output for two different infix expressions. The first expression is $(a+b)/(c-d)-(e*f)$, which is converted to the postfix expression $ab+cd-/ef*-$. The second expression is $((a+b-c+f*m)/n*k/(g+h))$, which is converted to the postfix expression $ab+c-fm+n/k*gh+/$. The program returns 0 (0x0) and the execution time is displayed for each run.