Institute of Computer Technology

B. Tech Computer Science and Engineering

Subject: DS (2CSE302)

**PRACTICAL-5**

**AIM: -** To learn applications of stack using ***infix*** to ***postfix*** conversion and ***postfix*** expression evaluation.

**1. Rohan is a 7th semester, who is studying at GUNI-ICT. During his “Compiler Design” course, his course faculty explained him that compiler work differently while it does evaluation of an expression due to below reasons:**

* **Infix expressions are readable and solvable by humans because of easily distinguishable order of operators, but compiler doesn't have integrated order of operators.**
* **Hence to solve the Infix Expression compiler will scan the expression multiple times to solve the sub-expressions in expressions orderly which is very in-efficient.**
* **To avoid this traversing, Infix expressions are converted to postfix expression before evaluation.**

**a) Write the c program to convert below infix expression into postfix using stack.**

**i. a-b\*c**

**ii. (a-b)\*c+(d+f)**

**Hint:**

* Infix expression can be represented with C+D, the operator is in the middle of the expression.
* In postfix expression, the operator will be at end of the expression, such as CD+
* Use isalnum() function, which checks whether the given character is alphanumeric or not. **isalnum**() function defined in **ctype.h** header file.
* Alphanumeric: A character that is either a letter or a number
* Postfix expression conversion

o **Input**: a-b\*c , **Output**: a b c \* -

o **Input**: (a-b)\*c+(d+f), **Output**: a b - c \* d f + +

***SOLUTION***

#include <stdio.h>

#include <ctype.h>

char Yash[100];

int top = -1;

void push(char a)

{

Yash[++top] = a;

}

char pop()

{

if (top == -1)

{

return -1;

}

else

{

return Yash[top--];

}

}

int priority(char a)

{

if (a == '(')

{

return 0;

}

if (a == '+' || a == '-')

{

return 1;

}

if (a == '\*' || a == '/')

{

return 2;

}

return 0;

}

int main()

{

char arr[100];

char \*e, x;

printf("Enter The Expression: ");

scanf("%s", arr);

e = arr;

printf("Postfix Expression: ");

while (\*e != '\0')

{

if (isalnum(\*e))

printf("%c ", \*e);

else if (\*e == '(')

push(\*e);

else if (\*e == ')')

{

while ((x = pop()) != '(')

printf("%c ", x);

}

else

{

while (priority(Yash[top]) >= priority(\*e))

printf("%c ", pop());

push(\*e);

}

e++;

}

while (top != -1)

{

printf("%c ", pop());

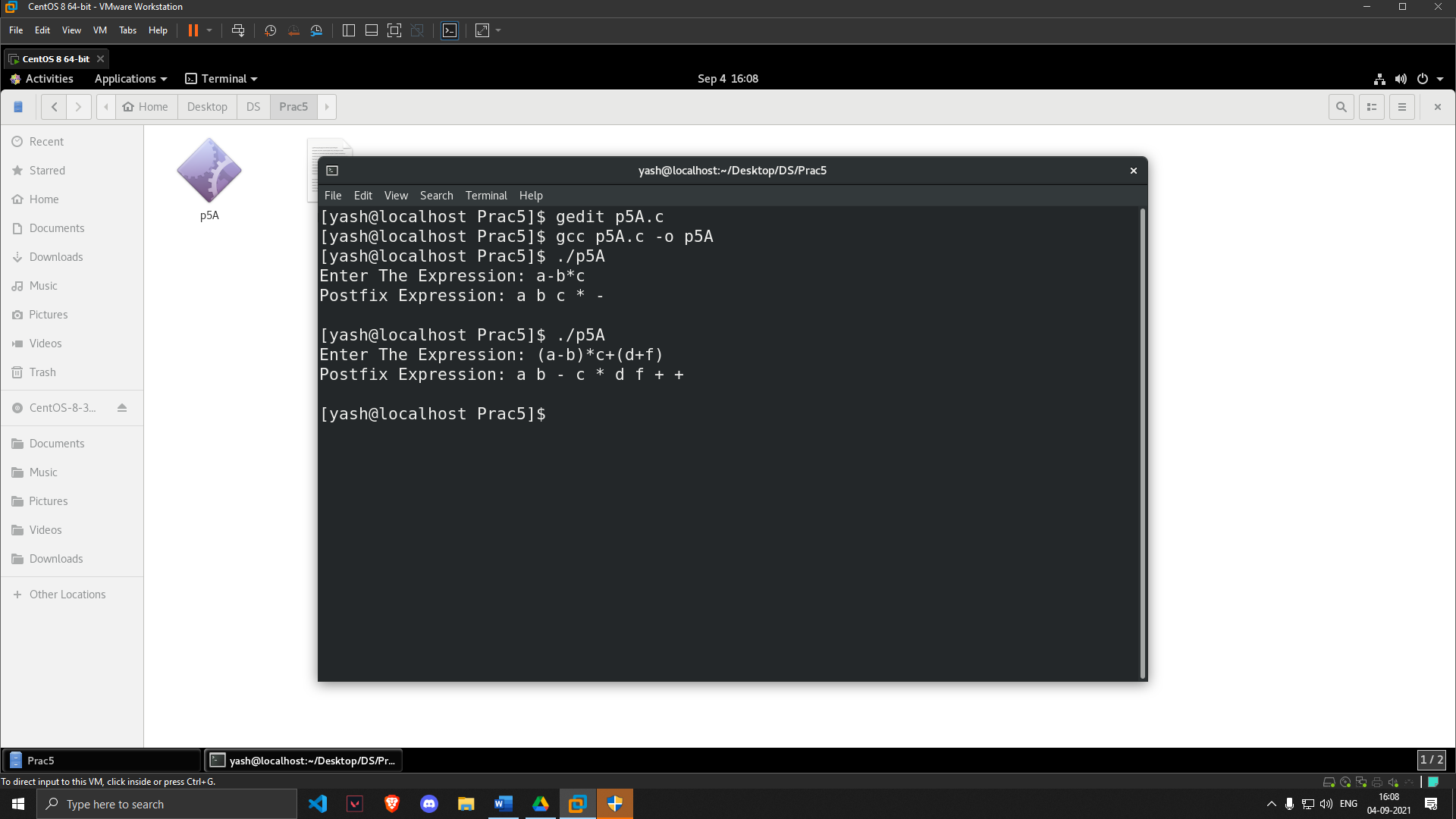
}

printf("\n\n");

return 0;

}

***OUTPUT***



**b) Rohan understood that why the conversion of the infix expression to postfix expression is important. Then, his friend Shyam asked him to evaluate the below postfix expression using stack using c program.**

**i. 237+\***

**ii. 53-8\*13+/**

**Hint:**

**•** Postfix expression evaluation

o **Input**: 237+\*, **Output**: 20

o **Input**: 53-8\*13+/, **Output**: 4

***SOLUTION***

#include<stdio.h>

#include<ctype.h>

int Yash[20];

int top = -1;

void push(int x)

{

Yash[++top] = x;

}

int pop()

{

return Yash[top--];

}

int main()

{

char arr[20];

char \*digit;

int diff=48;

int num1,num2,num3,num;

printf("Enter the expression: ");

scanf("%s",arr);

digit = arr;

while(\*digit != '\0')

{

if(isdigit(\*digit))

{

num = \*digit - diff;

push(num);

}

else

{

num1 = pop();

num2 = pop();

switch(\*digit)

{

case '+':

{

num3 = num1 + num2;

break;

}

case '-':

{

num3 = num2 - num1;

break;

}

case '\*':

{

num3 = num1 \* num2;

break;

}

case '/':

{

num3 = num2 / num1;

break;

}

}

push(num3);

}

digit++;

}

printf("\nThe result of expression %s = %d\n",arr,pop());

return 0;

}

***OUTPUT***

