Cycle 3

Malavika R Vikraman

CS7B – Roll No :31

Question 1

Develop an operator precedence parser for a given language.

Program :

```
#include<stdio.h>
#include<string.h>

#define MAX_SIZE 20

void main() {
    int numOfTerminals, top = -1, i, j, k, row, col;
    char terminals[10], operatorPrecedence[10][10], stack[MAX_SIZE],
input[MAX_SIZE];
    printf("Enter the no. of terminals: ");
    scanf("%d", &numOfTerminals);
    printf("Enter the terminals: \n");
    scanf("%s", terminals);

    //Operator precedence table
    printf("Enter the operator precedence table values: ");
    for(i = 0; i < numOfTerminals; i++)
        for(j = 0; j < numOfTerminals; j++) {
            printf("\nEnter the precedence value for %c %c: ",
terminals[i], terminals[j]);
            scanf(" %c", &operatorPrecedence[i][j]);
        }

    printf("\n------OPERATOR PRECEDeNCE TABLE------\n");
    for(i = 0; i < numOfTerminals; i++)
        printf("\t%c", terminals[i]);
    for(i = 0; i < numOfTerminals; i++) {
        printf("\n%c", terminals[i]);
        for(j = 0; j < numOfTerminals; j++) {
            printf("\t%c", operatorPrecedence[i][j]);
        }
    }

    //Parse input
    stack[++top] = '$';
    printf("\nInput the string to parse: ");
    scanf("%s", input);

    i = 0;
    printf("\nSTACK\t\tINPUT STRING\tACTION\n");
    printf("%s\t\t%s\t\t", stack, input);

    while(i <= strlen(input)) {
        for(k = 0; k < numOfTerminals; k++) {
            if(stack[top] == terminals[k]) {
                row = k;
                break;
            }
```

```
        }
        for(k = 0; k < numOfTerminals; k++) {
            if(input[i] == terminals[k]) {
                col = k;
                break;
            }
        }

        if((stack[top] == '$') && (input[i] == '$')) {
            printf("String is accepted!\n");
            break;
        }
        if((operatorPrecedence[row][col] == '<') ||
(operatorPrecedence[row][col] == '=')) {
            stack[++top] = operatorPrecedence[row][col];
            stack[++top] = input[i];
            printf("SHIFT %c", input[i]);
            i++;
        } else if(operatorPrecedence[row][col] == '>') {
            while(stack[top] != '<')
                --top;
            --top;
            printf("REDUCE");
        } else {
            printf("String is not accepted\n");
            break;
        }

        printf("\n");
        for(k = 0; k <= top; k++)
            printf("%c", stack[k]);

        printf("\t\t");
        for(k = i; k < strlen(input); k++)
            printf("%c", input[k]);
        printf("\t\t");
    }
}
```

Output :

```
File  Edit  View  Search  Terminal  Help
mec@cl1-1-1:~/CS7B$ touch pg1.c
mec@cl1-1-1:~/CS7B$ gcc pg1.c
mec@cl1-1-1:~/CS7B$ ./a.out
Enter the no. of terminals: 4
Enter the terminals:
+*i$
Enter the operator precedence table values:
Enter the precedence value for + +: >

Enter the precedence value for + *: <

Enter the precedence value for + i: <

Enter the precedence value for + $: >

Enter the precedence value for * +: >

Enter the precedence value for * *: >

Enter the precedence value for * i: <

Enter the precedence value for * $: >

Enter the precedence value for i +: >

Enter the precedence value for i *: >

Enter the precedence value for i i: =

Enter the precedence value for i $: >

Enter the precedence value for $ +: <

Enter the precedence value for $ *: <

Enter the precedence value for $ i: <

Enter the precedence value for $ $: A

------OPERATOR PRECEDeNCE TABLE------
        +       *       i       $
+       >       <       <       >
*       >       >       <       >
i       >       >       =       >
$       <       <       <       A
Input the string to parse: i*i+i$
```

```
Enter the precedence value for $ $: A

------OPERATOR PRECEDeNCE TABLE------
        +       *       i       $
+       >       <       <       >
*       >       >       <       >
i       >       >       =       >
$       <       <       <       A
Input the string to parse: i*i+i$

STACK           INPUT STRING    ACTION
$               i*i+i$          SHIFT i
$<i             *i+i$           REDUCE
$               *i+i$           SHIFT *
$<*             i+i$            SHIFT i
$<*<i           +i$             REDUCE
$<*             +i$             REDUCE
$               +i$             SHIFT +
$<+             i$              SHIFT i
$<+<i           $               REDUCE
$<+             $               REDUCE
$               $               String is accepted!
```