21BDS0340

Abhinav Dinesh Srivatsa

Compiler Design Lab

Assignment – III

Question 1

Aim:
Write a lex program to find out total number of vowels and consonants from a given string

Program:

```
%{
#include <stdio.h>

int vowelCount = 0;
int consonantCount = 0;
%}

%%
[aAeEiIoOuU]    { vowelCount++; }
[a-zA-Z]        { consonantCount++; }
.               ;

%%

int main() {
    yy_scan_string("Hello, World!");

    yylex();

    printf("Vowel count: %d\n", vowelCount);
    printf("Consonant count: %d\n", consonantCount);

    return 0;
}
```

Output:

```
(base) abhi@Abhinavs-MacBook-Pro Assignment 3 % lex letters.l
(base) abhi@Abhinavs-MacBook-Pro Assignment 3 % gcc -L/opt/homebrew/Cellar/flex/2.6.4_2/lib -lfl lex.yy.c
(base) abhi@Abhinavs-MacBook-Pro Assignment 3 % ./a.out
Vowel count: 3
Consonant count: 7
```

Question 2

Aim:
To convert an abstract syntax tree to machine code

Code:

```
#include <iostream>
using namespace std;

int num = 0;

void threeAddress(string input)
{
    // checking brackets
    string copy = input;
    reverse(copy.begin(), copy.end());
    for (int x = 0; x < input.length(); x++)
        if (input[x] == '(')
            for (int y = 0; y < input.length(); y++)
                if (copy[y] == ')')
                {
                    threeAddress(input.substr(x + 1, input.length() - y - 1 - x -
1));
                    input = input.substr(0, x) + "t" + to_string(num++) +
input.substr(input.length() - y);
                    copy = input;
                    reverse(copy.begin(), copy.end());
                    break;
                }

    // checking priority
    int op_index = 0;
    char op = '+';
    int op_count = 0;
    for (int x = 0; x < input.length(); x++)
    {
        if (input[x] == '+' || input[x] == '-' || input[x] == '*' || input[x] ==
'/')
            op_count++;
        if (input[x] == '/')
        {
            op_index = x;
            op = '/';
        }
        else if (input[x] == '*' && op != '/')
        {
            op_index = x;
            op = '*';
        }
        else if (input[x] == '-' && op != '/' && op != '*')
        {
```

```
            op_index = x;
            op = '-';
        }
        else if (input[x] == '+' && op != '/' && op != '*' && op != '-')
        {
            op_index = x;
            op = '+';
        }
    }
    if (op_count > 1)
    {
        int t1_n = 0, t2_n = 0;
        for (int x = op_index - 1; x >= 0; x--)
        {
            if (input[x] == '+' || input[x] == '-' || input[x] == '*' || input[x]
== '/' || input[x] == '=')
                break;
            t1_n = x;
        }
        for (int x = op_index + 1; x < input.length(); x++)
        {
            if (input[x] == '+' || input[x] == '-' || input[x] == '*' || input[x]
== '/' || input[x] == '=')
                break;
            t2_n = x;
        }
        string temp = input.substr(t1_n, t2_n - t1_n + 1);
        threeAddress(input.substr(t1_n, t2_n - t1_n + 1));
        input = input.substr(0, t1_n) + "t" + to_string(num++) + input.substr(t2_n
+ 1);
        threeAddress(input);
    }
    else
    {
        string token1 = "", token2 = "";
        for (int x = 0; x < input.length(); x++)
            if (input[x] == '+' || input[x] == '-' || input[x] == '*' || input[x]
== '/' || input[x] == '=')
            {
                token1 = token2;
                token2 = "";
                op = input[x];
            }
            else
                token2 += input[x];
        std::cout << "t" << num << " = " << token1 << " " << op << " " << token2 <<
"\n";
    }
}

int main()
```

```
{
    string input;
    cin >> input;
    threeAddress(input);
}
```

Output:

```
(a+b)-c/d+f
t0 = a + b
t1 = c / d
t2 = t0 - t1
t3 = t2 + f
```