Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

Course No. : CSE4130
Course Name : Formal Languages and Compilers Lab

Assignment No. : 05

Submitted By:

Name : Shahriar Hasan Chowdhury
ID No. : 17 01 04 030
Session : Spring - 2020
Section : A (A2)

Assignment Questions:

Problem 1: Implement the following CFG in the way shown above.

A → aXd X → bbX X → bcX X → ε

Answer Question:

```
/**
Implementation of the following CGF

A -> aXd
X -> bbX
X -> bcX
X -> d

**/

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

int i = 0;

int X(char str[], int length)
{
    if(i == length -1)
        return 1;

    else if(str[i] == 'b' && str[i+1] == 'b') {
        i = i+2;
        X(str, length);
    }
    else if(str[i] == 'b' && str[i+1] == 'c') {
        i = i+2;
        X(str, length);
    }
    else
        return 0;
}


int A(char str[], int len)
{
    if(str[0] != 'a') {
        return 0;
    }
    else {
        i++;
        if(X(str, len)) {
            if(str[strlen(str)-1] == 'd')
```

```
                return 1;
            else
                return 0;
        }
    }
}

void CFG(char str[], int len)
{

    if(A(str, len)) {
        printf("[+] %s is Accepted by the given CGF\n", str);
    }
    else {
        printf("[-] %s is rejected by the given CGF\n", str);
    }
}

int main()
{
    char str[1000];
    printf("--> ");
    while(scanf("%[^\n]%*c", str) != EOF){
        i = 0;
        int len = strlen(str);

        CFG(str, len);
        printf("--> ");
    }
}
```

Problem 2: Implement the CFG shown above for generating simple arithmetic expressions.

Answer Question:

```
/**
Implementation of the following CGF

<Exp> -> <Term> + <Term> | <Term> - <Term> | <Term>
<Term> -> <Factor> * <Factor> | <Factor> / <Factor> | <Factor>
<Factor> -> ( <Exp> ) | ID | NUM

    ID -> a|b|c|d|e
    NUM -> 0|1|2|3..|9

**/


#include <stdio.h>
```

```c
#include <stdlib.h>
#include <string.h>


char str[1000];

int factor(int ci,int li);
int term(int ci, int li);
int expressions(int ci,int li);
int isId(int i);
int isNum(int i);
void CFG(char str[]);


int isId(int i){
    if(str[i]=='a' || str[i]=='b' || str[i]=='c' || str[i]=='d' || str[i]==
'e')
        return 1;
    return 0;
}

int isNum(int i){
    if(isdigit(str[i]))
        return 1;
    return 0;
}

int term(int ci, int li)
{
    ///printf("%c +++ %c \n", str[ci], str[li]);
    int i=ci;

    while(i<li && str[i]!='*' && str[i]!='/')
        ++i;
    ///printf("+++ %c\n", str[i]);
    if(i==li)
        return factor(ci,li);
    else if((str[i]=='*' || str[i]=='/') && factor(ci,i-1))
        return factor(i+1,li);
    else
        return 0;
}

int expressions(int ci,int li)
{
    ///printf("%c ---- %c \n", str[ci], str[li]);
    int i=ci;

    while(i<li && str[i]!='+' && str[i]!='-'&& str[i]!='*' && str[i]!='/')
        ++i;
```

```
    if(ci==li)
        return term(ci,li);
    else if((str[i]=='+' || str[i]=='-
'|| str[i]=='*'|| str[i]=='/') && term(ci,i-1))
        return term(i+1,li);
    else
        return 0;
}
int factor(int ci,int li){
    ///printf("%c === %c \n", str[ci], str[li]);
    if(isId(ci))
        return 1;
    else if(isNum(ci))
        return 1;

    else if(str[ci] == '(' && str[li] == ')')
        return expressions(ci+1,li-1);
    else
        return 0;
}
void CFG(char str[]){
    int len = strlen(str);
    int ci = 0;
    int li = len-1;
    if(expressions(ci,li))
        printf("[+] %s is Accepted by the given CGF\n", str);
    else
        printf("[-] %s is rejected by the given CGF\n", str);
}

int main(){
    printf("--> ");
    while(scanf("%[^\n]%*c", str) != EOF){
        CFG(str);
        printf("--> ");
    }
    return 0;
}
```