Ahsanullah University of Science and Technology
Department of Computer Science & Engineering
Semester Spring 2020

CSE 4130
Formal Languages and Compilers Lab

Assignment 5

Submitted To

Md. Aminur Rahman
Assistant Professor, CSE, AUST
Saaduddin Mahmud
Lecturer, CSE, AUST

Submitted By

Md Siam Islam
ID: 17.01.04.124
Lab Group: C1

Lab Exercise-1

Implement the following CFG in the way shown above.

A → aXd
X → bbX
X → bcX
X → ϵ

Code:

```c
#include<stdio.h>

char str[100];
int i=0,l,f;
void X() {
    if (str[i] == 'b' && (str[i+1] == 'b' || str[i+1]=='c')) {
        i+=2;
        f=1;
    }
    else {f=0; return;}
    if (i<l-1) X();
}
void A() {
    if (str[i++] == 'a'){
        if(i!=l-1)
            X();
        else if(str[i]=='d'){
            f=1;
            return;
        }
        if(f){
            if(str[i]!='d')
                f=0;
        }
    }
    return;
}

int main(){
    printf("Enter string: ");
    scanf("%s", str);
    l=strlen(str);
    A();
    if(f==1)
        printf("Valid\n");
    else
        printf("Invalid\n");
    return 0;
}
```

Lab Exercise-2

Implement the CFG shown below for generating simple arithmetic expressions.

⟨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 | ... | 9

Non-terminal symbols: ⟨Exp⟩, ⟨Term⟩, ⟨Factor⟩

Terminal symbols: +, -, *, /, (,), a, b, c, d,e, 0, 1, 2, 3, ..., 9

Start symbol: ⟨Exp⟩

Code:

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

char str[100];
int f,l,i=0;
void Factor()
{
    if(isdigit(str[i]) || (str[i] >= 'a' && str[i] <= 'e'))
    {
        i++;
        f = 1;
        return;
    }
    else if (str[i] == '(')
    {
        i++;
        Exp();
        if(str[++i] == ')')
        {
            f = 1;
            return;
        }
    }
}
void Term()
{
    Factor();
    if(i<l-1)
    {
        if(str[i] == '*' || str[i] == '/')
```

```
        {
            i++;
            Factor();
        }
    }
    else if(f == 1)
        return;
}
void Exp()
{
    Term();
    if(i<l-1)
    {
        if(str[i] == '+' || str[i] == '-')
        {
            i++;
            Term();
        }
    }
    else if(f == 1)
        return;

}
int main(void)
{
    printf("Enter simple arithmetic expressions: ");
    scanf("%s", str);
    l = strlen(str);
    if(l>=1)
        Exp();
    else
        printf("Empty!\n");
    if(i == l && f == 1)
        printf("Valid Expression\n");
    else
        printf("Invalid Expression\n");

    return 0;
}
```

Assignment

Implement the following grammar in C.

⟨stat⟩→⟨asgn_stat⟩|⟨dscn_stat⟩|⟨loop_stat⟩
⟨asgn_stat⟩→id=⟨expn⟩
⟨expn⟩→⟨smpl_expn⟩ ⟨extn⟩
⟨extn⟩→⟨relop⟩ ⟨smpl_expn⟩| ϵ
⟨dcsn_stat⟩→if(⟨expn⟩)⟨stat⟩ ⟨extn1⟩
⟨extn1⟩→ else⟨stat⟩|ϵ
⟨loop_stat⟩→while(⟨expn⟩) ⟨stat⟩|for(⟨asgn_stat⟩;⟨expn⟩; ⟨asgn_stat⟩)⟨stat⟩

⟨relop⟩→ ==|!=|⟨=|⟩=|⟩|⟨
Note: ⟨smpl_expn⟩ can be implemented using the materials demonstrated in this session.

Code:

```
#include<stdio.h>
#include<string.h>
#include <stdbool.h>
int f,i=0,l,s=0;
char str[100];
void stat();
void asgn_stat();
void dscn_stat();
void loop_stat();
bool relop();
void expn();
void extn();
void smpl_expn();
void Term();
void Factor();
int main()
{
    printf("Enter a string:\n");
    scanf("%s", str);
    l = strlen(str);
    printf("\nOutput: ");
    if (l>=1)
        stat();
    else
        printf("Invalid String\n");
    if (l == i && f )
        printf("Valid String\n");
    else
        printf("Invalid String\n");
    return 0;
}

void stat()
{
    asgn_stat();
    if(s==0)
        dscn_stat();
    else if(s==0)
        loop_stat();
}
void asgn_stat()
{
    if(i<l && (str[i]>='a' && str[i]<='e'))
    {
        if(str[++i]=='=')
        {
            i++;
            expn();
        }
        s=1;
```

```
    }
}
void expn()
{
    smpl_expn();
    if(f)
        extn();
}
bool relop()
{
    if(f && i<l && (str[i]=='=' || str[i]=='!' || str[i]=='<' || str[i
   ]=='>'))
    {
        i++;
        if(i<l && str[i]=='=')
        {
            i++;
            return true;
        }
    }
    return false;
}
void extn()
{
    if(f && i<l && relop())
        smpl_expn();
}
void extn1()
{
    if(f && i+3<l && str[i]=='e' && str[i+1]=='l' && str[i+2]=='s' &&
   str[i+3]=='e')
    {
        i=i+4;
        stat();
    }
}
void dscn_stat()
{
    if(i<l && str[i]=='i' && str[i+1]=='f')
    {
        i+=2;
        if(str[i]=='(')
        {
            i++;
            expn();
            if(f && i<l && str[i]==')')
            {
                i++;
                stat();
                if(f)
                    extn1();
            }
            else f=0;
        }
        else f=0;
    }
    s=1;
}
```

```
107 void loop_stat()
108 {
109     if(i<l && str[i]=='w' && str[i+1]=='h' && str[i+2]=='i' && str[i
    +3]=='l' && str[i+4]=='e')
110     {
111         i+=5;
112         if(str[i]=='(')
113         {
114             i++;
115             expn();
116             if(f && i<l && str[i]==')')
117             {
118                 i++;
119                 stat();
120             }
121             else f=0;
122         }
123         else f=0;
124     }
125     else if(i<l && str[i]=='f' && str[i+1]=='o' && str[i+2]=='r')
126     {
127         i+=3;
128         if(str[i]=='(')
129         {
130             i++;
131             asgn_stat();
132             if(f && i<l && str[i]==';')
133             {
134                 i++;
135                 expn();
136                 if(f && i<l && str[i]==';')
137                 {
138                     i++;
139                     asgn_stat();
140                     if(f && i<l && str[i]==')')
141                     {
142                         i++;
143                         stat();
144                     }
145                     else f=0;
146                 }
147                 else f=0;
148             }
149             else f=0;
150         }
151         else f=0;
152     }
153 }
154 void smpl_expn()
155 {
156     Term();
157     if(f && i<l &&(str[i]=='+' || str[i]=='-'))
158     {
159         i++;
160         Term();
161     }
162 }
163 void Term()
```

```
{
    Factor();
    if(f && i<l && (str[i]=='*' || str[i]=='/'))
    {
        i++;
        Factor();
    }
}
void Factor()
{
    if(i<l && str[i]=='(')
    {
        i++;
        f=1;
        smpl_expn();
        if(f && str[i]==')')
            i++;
        else
            f=0;
    }
    else if(i<l && ((str[i]>='a' && str[i]<='e')||isdigit(str[i])))
    {
        i++;
        f=1;
    }
    else f=0;
}
```