Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

CSE 4130

Formal languages and Compilers lab

Assignment No: 05

Submitted By:

Name: Anika Tanzim

ID: 16.02.04.072

Group: B1

Date of Submission: **16 September 2020**

1. **Implement the following CFG, in the way shown above, or with the help of a user-defined stack, that is, in the way it may work in an implementation of the transition function of a PDA. A → aXd X → bbX X → bcX X → epsilon**

Code:

```
/*A → aXd
X → bbX
X → bcX
X →  */

#include<stdio.h>

char str[10];
int i=0;
int l=0;
int f=1;

void A(void);
void X(void);


void A(){
   if(str[i]=='a'){

      i++;
      X();

      if(f==1){
         if(str[i]=='d'){
               f=2;
               return;
         }
         else{
            f=0;
            return;
         }

      }
```

```
    }else
    {
        f=0;
        return;
    }

}

void X(){
    if(i==l-1){ // checking epsilon
        f=1;
        return;
    }
    else if(str[i]== 'b'){
        i++;
        if(str[i]=='b' || str[i]== 'c'){
            i++;
            X();
        }
    }

}

int main()
{

    printf("CFG: \n");
    printf("\tA -> aXd\n\tX -> bbX\n\tX -> bcX\n\tX -> epsilon \n");
    printf("\nEnter a string to parse: ");
    scanf("%s",&str);

    l=strlen(str);
    //printf("%d",l);

    if(l>=2){
        A();
    }
    else{
        printf("Invalid String");
```

```
    }

    if(f==0){
        printf("Invalid String");
    }
    else if(f==2 && i==l-1){
        printf("Valid String");
    }


    return 0;
}
```

Output:

```
CFG:
        A -> aXd
        X -> bbX
        X -> bcX
        X -> epsilon

Enter a string to parse: abbbcbcd
Valid String


CFG:
        A -> aXd
        X -> bbX
        X -> bcX
        X -> epsilon

Enter a string to parse: asd
Invalid String
```

2. **Implement the CFG shown for generating simple arithmetic expressions.**

Code:

```
/*<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
*/

#include<stdio.h>

char str[10];
int i=0;
int l;
int f=0;

void E(void);
void T(void);
void F(void);


void E()
{
   T();
   if(f && i<l &&(str[i]=='+' || str[i]=='-'))
   {
      i++;
      T();
   }

}
void T()
{
   F();
   if(f && i<l && (str[i]=='*' || str[i]=='/'))
   {
      i++;
```

```
        F();
    }

}

void F()
{
    if(i<l && str[i]=='(')
    {
        i++;
        f=1;
        E();
        if(f && str[i]==')')
        {
            i++;
        }
        else
        {
            f=0;
        }

    }
    else if(i<l && (str[i]=='a' || str[i]=='b'|| str[i]=='c' || str[i]=='d' || str[i]=='e'))
    {
        i++;
        f=1;
    }
    else if(i<l && isdigit(str[i])==1)
    {
        i++;
        f=1;
    }
    else f=0;
}
int main(void) {

            printf("CFG: \n");
    printf("\tE -> T+T | T-T | T \n\tT -> F*F | F/F | F \n\tID -> a|b|c|d|e \n\tNUM-> 0|1|2|...|9 \n");
    printf("\nEnter a string to parse: ");
            scanf("%s", str);
```

```
        l = strlen(str);

    if (l>=1)
        E();
    else
                printf("\nInvalid String\n");

    if (l == i && f )
                printf("\nValid String\n");
        else
                printf("\nInvalid String\n");

    return 0;
}
```

Output:

CFG:

E -> T+T | T-T | T
T -> F*F | F/F | F
ID -> a|b|c|d|e
NUM-> 0|1|2|...|9

Enter a string to parse: a*b+(1-2)

Valid String

3.Implement the following grammar in C

Code:

```
#include<stdio.h>
#include<string.h>
#include <stdbool.h>
int f=0;
int i=0;
int l;
char str[10];
int s=0;

void stat(void);
void asgn_stat(void);
void dscn_stat(void);
void loop_stat(void);
bool relop(void);
void expn(void);
void extn(void);

void E(void);
void T(void);
void F(void);

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]=='b'|| str[i]=='c' || str[i]=='d' || str[i]=='e'))
```

```
    {
        i++;
        if(str[i]=='=')
        {
            i++;
            expn();
        }
        s=1;
    }
}

void expn()
{
    E();
    if(f)
    {
        extn();
    }
}

bool relop()
{
    if(f && i<l && (str[i]=='=' || str[i]=='!'))
    {
        i++;
        if(i<l && str[i]=='=')
        {
            i++;
            return true;
        }


        else return false;
    }
    else if(f && i<l && (str[i]=='<' || str[i]=='>'))
    {
        i++;
        if(i<l && str[i]=='=')
        {
            i++;
```

```
        }

        return true;
    }
    return false;
}



void extn()
{
    if(f && i<l && relop())
    {
        E();
    }
}









void extn1()
{
    if(f && i+3<l && str[i]=='e' && str[i+1]=='l' && str[i+2]=='s' && str[i+3]=='e') //f already 1
    {
        i=i+4;
        stat();
    }
}
void dscn_stat()
{
    if(i<l && str[i]=='i' && str[i+1]=='f')
    {
        i=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;
}

void loop_stat()
{

  if(i<l && str[i]=='w' && str[i+1]=='h' && str[i+2]=='i' && str[i+3]=='l' && str[i+4]=='e')
  {
     i=i+5;
     if(str[i]=='(')
     {
        i++;
        expn();
        if(f && i<l && str[i]==')')
        {
           i++;
           stat();
        }
        else f=0;
     }
     else f=0;
  }
  else if(i<l && str[i]=='f' && str[i+1]=='o' && str[i+2]=='r')
  {

     i=i+3;
     if(str[i]=='(')
```

```
        {
          i++;
          asgn_stat();
          if(f && i<l && str[i]==';')
          {
             i++;
             expn();
             if(f && i<l && str[i]==';')
             {
                i++;
                asgn_stat();
                if(f && i<l && str[i]==')')
                {
                   i++;
                   stat();
                }
                else f=0;


             }
             else f=0;

          }
          else f=0;
        }
        else f=0;
    }


    }
```

```
void E()
{
   T();
   if(f && i<l &&(str[i]=='+' || str[i]=='-'))
```

```
    {
        i++;
        T();
    }

}
void T()
{
    F();
    if(f && i<l && (str[i]=='*' || str[i]=='/'))
    {
        i++;
        F();
    }

}

void F()
{
    if(i<l && str[i]=='(')
    {
        i++;
        f=1;
        E();
        if(f && str[i]==')')
        {
            i++;
        }
        else
        {
            f=0;
        }

    }
    else if(i<l && (str[i]=='a' || str[i]=='b'|| str[i]=='c' || str[i]=='d' || str[i]=='e'))
    {
        i++;
        f=1;
    }
    else if(i<l && isdigit(str[i])==1)
```

```
    {
       i++;
       f=1;
    }
    else f=0;
}


int main()
{


    printf("\n\t<stat> -> <asgn_stat> | <dscn_stat> | <loop_stat>\n\t<asgn_stat> ->id =
<expn>\n\t<expn>  -><smpl_expn> <extn>\n\t<extn>  -><relop> <smpl_expn> |
epsilon\n\t<dcsn_stat>  -> if (<expn> ) <stat> <extn1>\n\t<extn1>  -> else <stat> |
epsilon\n\t<loop_stat>→while (<expn>) <stat>for (<asgn_stat> ; <expn> ; <asgn_stat> )
<stat>\n\t<relop>  -> == | != | <= | >= | > | <");

    printf("\nEnter a string: ");
    scanf("%s", str);
          l = strlen(str);
    if (l>=1)
    {
       stat();
    }
    else
       printf("\nInvalid String\n");
    if (l == i && f )
       printf("\nValid String\n");
    else
       printf("\nInvalid String\n");
    return 0;
}
```

Output1:

// <asgn_stat> implemented

<stat> -> <asgn_stat> | <dscn_stat> | <loop_stat>
<asgn_stat> ->id = <expn>
<expn> -><smpl_expn> <extn>
<extn> -><relop> <smpl_expn> | epsilon
<dcsn_stat> -> if (<expn>) <stat> <extn1>
<extn1> -> else <stat> | epsilon
<loop_stat>ΓåÆwhile (<expn>) <stat>∩é╨ for (<asgn_stat> ; <expn> ; <asgn_stat>) <stat>
<relop> -> == | != | <= | >= | > | <

Enter a string: a=a+b<=a*b

Valid String

Process returned 0 (0x0)   execution time : 17.835 s
Press any key to continue.

Output2:

// <dcsn_stat> implemented

<stat> -> <asgn_stat> | <dscn_stat> | <loop_stat>
<asgn_stat> ->id = <expn>
<expn> -><smpl_expn> <extn>
<extn> -><relop> <smpl_expn> | epsilon
<dcsn_stat> -> if (<expn>) <stat> <extn1>
<extn1> -> else <stat> | epsilon
<loop_stat>ΓåÆwhile (<expn>) <stat>∩é╨ for (<asgn_stat> ; <expn> ; <asgn_stat>) <stat>
<relop> -> == | != | <= | >= | > | <

Enter a string: if(a)aelseb

Valid String

Process returned 0 (0x0)   execution time : 28.113 s
Press any key to continue.

Output3:

//<loop_stat> implemented but invalid

<stat> -> <asgn_stat> | <dscn_stat> | <loop_stat>
<asgn_stat> ->id = <expn>
<expn> -><smpl_expn> <extn>
<extn> -><relop> <smpl_expn> | epsilon
<dcsn_stat> -> if (<expn>) <stat> <extn1>
<extn1> -> else <stat> | epsilon
<loop_stat>ΓåÆwhile (<expn>) <stat>∩é╜ for (<asgn_stat> ; <expn> ; <asgn_stat>) <stat>
<relop> -> == | != | <= | >= | > | <

Enter a string: while(a)a=a+1

Invalid String

Process returned 0 (0x0)   execution time : 9.223 s
Press any key to continue.