Ahsanullah University of Science and Technology (AUST)

Department of Computer Science and Engineering

Course No : CSE4130

Course Title: Formal Languages and Compilers Lab

Session : Spring 2020

Assignment No : 05

Submitted By :

Lab Group : A1

Name : Umme Habiba

ID :170104004

Question-1: Implement the following CFG in the way shown above.

A → aXd
X → bbX
X → bcX
X → epsilon

Answer:

```
#include<stdio.h>

#include<string.h>

void A(void);

void X(void);

char str[100];

int f=0;

int i=0;

int l;

int main(void) {
   printf("Enter a string to Parse: ");
   scanf("%s",&str);
   l=strlen(str);
   if(l>=1)
      A();
   else
      printf("\nInvalid String\n");
   if(l==i && f)
      printf("\nValid String\n");
   else
      printf("\nInvalid String\n");
   return 0;
```

```
}
void A(){
   if(str[i]=='a') {
      i++;
      X();
      if(f && str[i]=='d') {
         f=1;
         return;
      }
   }
   else{
      f=0;
      return;
   }
}
void X(){
   if((l-1)==i) {
      f=1;
      i++;
      return;
   }
   else{
      if (str[i] == 'b') {
         i++;
         if(str[i]=='b'||str[i]=='c') {
            i++;
```

```
            X();
          }
        }
        else{
          f=0;
          return;
        }
      }
}
```

Question-2: Implement the CFG shown above 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

Answer:

```
#include<stdio.h>
#include<string.h>
void Exp(void);
void term(void);
void fact(void);
void id(void);
void num(void);
char str[100];
int f=0;
int i=0;
int l;
```

```
void Exp(){
   term();
   if(f && (str[i]=='+'||str[i]=='-')) {
      i++;
      term();
   }
}
void term(){
   fact();
   if(f && (str[i]=='*'||str[i]=='/')) {
      i++;
      fact();
   }
}
void fact(){
   if(i<l && str[i]=='(') {
      i++;
      f=1;
      Exp();
      if(f && str[i]==')')
         i++;
      else{
         f=0;
         return;
      }
   }
```

```
    else{
        if(i<l && (str[i]=='0'||str[i]=='1'||str[i]=='2'||str[i]=='3'||str[i]=='4'||
                str[i]=='5'||str[i]=='6'||str[i]=='7'||str[i]=='8'||str[i]=='9')) {
            i++;
            f=1;
            return;
        }
        else if(i<l && (str[i]=='a'||str[i]=='b'||str[i]=='c'||str[i]=='d'||str[i]=='e')) {
           i++;
           f=1;
           return;
        }
        else{
            f=0;
            return;
        }
    }
}
int main(void) {
   printf("Enter a string to Parse: ");
   scanf("%s",&str);
    l=strlen(str);
    if(l>=1)
        Exp();
    else
        printf("\nInvalid String\n");
```

```
    if(l==i && f)
        printf("\nValid String\n");
    else
        printf("\nInvalid String\n");
    return 0;
}
```

Question-3: Implement the following grammar in C.

<stat>→<asgn_stat> epsilon <dscn_stat> epsilon <loop_stat>
<asgn_stat>→id = <expn>
<expn>→<smpl_expn> <extn>
<extn>→<relop> <smpl_expn> | epsilon
<dcsn_stat>→ if (<expn>) <stat> <extn1>
<extn1>→ else <stat> | e
<loop_stat>→while (<expn>) <stat>efor (<asgn_stat> ; <expn> ; <asgn_stat>) <stat>
<relop>→ == epsilon!= epsilon <= epsilon >= epsilon > epsilon <

Answer:

```
#include<stdio.h>
#include<string.h>
#include<stdbool.h>
bool stat(char *input);
bool loop_stat(char *input);
bool dcsn_stat(char *input);
bool asgn_stat(char *input);
bool expn(char *input);
bool smpl_expn(char *input);
bool extn(char *input);
bool term(char *input);
bool factor(char *input);
bool id(char *input);
```

```
bool num(char *input);
bool extn1(char *input);
int i;
int main(){
   char input[100];
   printf("Enter a string to Parse: ");
   scanf("%s",&input);
   if(stat(input)){ printf("\nValid String\n"); }
   else{ printf("\nInvalid String\n");}
   return 0;
}
bool asgn_stat(char *input) {
   char newx[100]; int c = 0;
   char x[5];
   x[0] = input[0];//first char
   if(id(x)){//id check
      if(input[1] == '='){
         for( i=2; i<strlen(input); i++){
            newx[c++] = input[i];//copy full length
         }
         newx[c] = '\0';
         if(expn(newx)){
            return true;
         }
      }
   }
```

```
  return false;
}
bool dcsn_stat(char *input) {
  char x[100], y[100], z[100]; int c1 = 0, c2 =0, c3 = 0;
  if(input[0] == 'i' && input[1] == 'f' && input[2] == '('){
    int i = 3;
    while(1){
      x[c1++] = input[i];
      i++;
      if(input[i] == ')'){break;} }
    x[c1] = '\0';
    i++;
    if(expn(x)){
      while(1){
        y[c2++] = input[i];
        i++;
        if(input[i] == 'e' && input[i+1] == 'l' && input[i+2] == 's' && input[i+3] == 'e'){break;} }
      y[c2] = '\0';
      if(stat(y)){
        while(1){
          z[c3++] = input[i];
          i++;
          if(i >= strlen(input)){break;}
        }
        z[c3] = '\0';
        if(extn1(z)){
```

```
            return true; }}}
  }
  return false;
}
bool loop_stat(char *input) {
  char x[100], y[100], z[100], p[100]; int c1 = 0, c2 = 0, c3 =0, c4 = 0;
  if(input[0] == 'w' && input[1] == 'h' && input[2] == 'i' && input[3] == 'l' && input[4] == 'e' &&
input[5] == '('){
    int i = 6;
    while(1){
      x[c1++] = input[i];
      i++;
      if(input[i] == ')'){break;} }
    x[c1] = '\0';
    i++;
    if(expn(x)){
      while(1){
        y[c2++] = input[i];
        i++;
        if(i >= strlen(input)){break;}
      }
      y[c2] = '\0';
      if(stat(y)){return true;} }
  }
  else if(input[0] == 'f' && input[1] == 'o' && input[2] == 'r' && input[3] == '('){
    int i = 4;
    while(1){
```

```
   x[c1++] = input[i];
   i++;
   if(input[i] == ';'){break;} }
x[c1] = '\0';
i++;
if(asgn_stat(x)){
   while(1){
      y[c2++] = input[i];
      i++;
      if(input[i] == ';'){break;}
   }
   i++;
   y[c2] = '\0';
   if(expn(y)){
      while(1){
         z[c3++] = input[i];
         i++;
        if(input[i] == ')'){break;}
      }
      i++;
      z[c3] = '\0';
      if(asgn_stat(z)){
         while(1){
            p[c4++] = input[i];
            i++;
            if(i>=strlen(input)){break;}
```

```
            }
            p[c4] = '\0';
            if(stat(p)){return true;}
          }
        }
      }
    }
    else{
      return false;
    }
}
bool expn(char *input) {
    char x[100], y[100]; int c1 = 0, c2 = 0, turn = 1;
    for( i=0; i<strlen(input); i++){
      if( (input[i] == '=' && input[i+1] == '=') || (input[i] == '!' && input[i+1] == '=') || (input[i] == '>' && input[i+1] == '=') || (input[i] == '<' && input[i+1] == '=') ){
        turn = 2;
      }
      else if(input[i] == '>' || input[i] == '<'){
        turn = 2;
      }
      if(turn == 1){
        x[c1++] = input[i];
      }
      else if(turn == 2){
        y[c2++] = input[i];
      }
```

```
  }
  x[c1] = '\0';
  y[c2] = '\0';
  if(smpl_expn(x) && extn(y)){ return true;}
  else{ return false;}
}
bool smpl_expn(char *input)
{
  char x[100], y[100]; int c1 = 0, c2 = 0, turn = 1;
  for( i=0; i<strlen(input); i++){
    if( input[i] == '+' || input[i] == '-' ){
      turn = 2;
      i++;
    }
    if(turn == 1){
      x[c1++] = input[i];
    }
    if(turn == 2){
      y[c2++] = input[i];
    }
    else if(input[i] == '*' || input[i] == '/'){break;}
  }
  x[c1] = '\0';
  y[c2] = '\0';
  if(strlen(y) > 0){
    if(term(x) && term(y)){ return true;}
```

```
    else{ return false;}
  }
  else{
    if(term(x)){ return true;}
    else{ return false;}
  }
}
bool extn(char *input) {
  char x[100]; int c1 = 0;
  if(strlen(input) == 0){return true;}
  if( (input[0] == '=' && input[1] == '=') || (input[0] == '!' && input[1] == '=') || (input[0] == '>' &&
input[1] == '=') || (input[0] == '<' && input[1] == '=')){
    for( i=2; i<strlen(input); i++){
      x[c1++] = input[i];
    }
  }
  else if(input[0] == '>' || input[0] == '<'){
    for( i=1; i<strlen(input); i++){
      x[c1++] = input[i];
    }
  }
  x[c1] = '\0';
  if(smpl_expn(x)){ return true;}
  return false;
}
bool term(char *input) {
  char x[100], y[100]; int c1 = 0, c2 = 0, turn = 1;
```

```
  for( i=0; i<strlen(input); i++){
    if( input[i] == '*' || input[i] == '/' ){
      turn = 2;
      i++;
    }
    if(turn == 1){
      x[c1++] = input[i];
    }
    if(turn == 2){
      y[c2++] = input[i];
    }
    else if(input[i] == '+' || input[i] == '-'){break;}
  }
  x[c1] = '\0';
  y[c2] = '\0';
  if(strlen(y) > 0){
    if(factor(x) && factor(y)){ return true;}
    else{ return false;}
  }
  else{
    if(factor(x)){ return true;}
    else{ return false;}
  }
}
bool factor(char *input) {
  if(strlen(input) > 1){
```

```
    if(smpl_expn(input)){return true;}
  }
  if(id(input) || num(input)){return true;}
  else{return false;}
}
bool id(char *input) {
  if(input[0] == 'a' || input[0] == 'b' || input[0] == 'c' || input[0] == 'd' || input[0] == 'e'){ return true;}
  else{ return false;}
}
bool num(char *input) {
  if(input[0] >= '0' && input[0] <='9'){return true;}
  else{ return false;}
}
bool stat(char *input) {
  if(asgn_stat(input) || dcsn_stat(input) || loop_stat(input)){ return true; }
  else{ return false;}}
bool extn1(char *input) {
  char x[100], y[100]; int c1 = 0, c2 = 0;
  if(strlen(input) == 0){ return true;}
  for( i=4; i<strlen(input); i++){
    x[c1++] = input[i];
  }
  x[c1] = '\0';
  if(stat(x)){ return true;}
  return false;
}
```