Term project #2.

김효수 교수님
20166450 김영민
Deadline 2019.06.09

■input output explain(not all of test ouput just important things)

1.

test.c

```
int func(int a, char b){
    char test;
    int c = -11112;

    return 0;
}
```

This input is needed to show that -11112 is an integer. That means '-' is used as a operator. I already solved this problem in term project 1. But here the problem is 'int c = -11112;' It doesn't define in the given CFG.

test_prj2.out

```
GRAMMAR : REJECTED
STACK: ['0', 'vtype', '1', 'identifier', '5', 'lparen', '8', 'vtype', '10', 'identifier', '13', 'comma', '15', 'vtype', '
Index Error!!
```

Therefore, the result is 'rejected '

```
, '41', 'rparen', '17', 'lbrace', '29', 'STMT', '22', 'vtype', '19', 'identifier', '31']<- Error occured from here.
```

You can see where the Error occur.

2.

test11.c

```
int foo(int a, char b){
    char c;
    while(char c = 1){
        a = b+1;
    }
    return 0;
}
```

this is going to be reject. Because inside while loop, it must be condition. 'char c = 1' is not condition.

test11_pj2.out

```
GRAMMAR : REJECTED
STACK: ['0', 'vtype', '1', 'identifier', '5', 'lparen', '8', 'vtype', '10', 'identifier', '13', 'comma', '15', 'vtype', '
Index Error!!

ifier', '41', 'rparen', '17', 'lbrace', '29', 'STMT', '22', 'while', '21', 'lparen', '33']<- Error occured from here.
```

so, Grammar rejected and error occur at while (<-

3. test12.c

```
int foo(int a, char b){
    char c;
    while(a == b){
        a = b+1;
    }
    return 0;
}
```

This is different from previous test11.c. I used condition in the while(a == b). So the result will be 'accepted'

4. test3.c

```
int foofunc(int a, int b) {
    int c ;
    c = a - b;
    c = a + b;
    c = a * b;
    c = a / b;
    return 0;
}
```

This is for all of operator. Following the CFG, It's going to be 'Accepted'

```
GRAMMAR : ACCEPTED
```

■Code explain

First of all, input terminals, nonterminals, parsing_table, Starting Symbol, Grammar, Parsing_table which I have already computed by hands.
Secondly, defind method of First.
Third, defind method of Follow.
Fourth, compute the grammar using First and Follow.

⟨Method First⟩

```
def First(X):
    if X in terminals:
        return {X}
    else:
        #make list for a second
        first_list = []

        #set set.
        first = set([])

        #append nonterminal in first_list.
        first_list.append(X)
```

Compute X of First set.
If x is nonterminal, set Set.

```
        for prod in Grammar_dic[X]:
            #for head
            if prod == ['^']:
                first.add('^')

            else:
                for (i, symbol) in enumerate(prod.split()):
                    if symbol not in first_list:
                        symbol_first = First(symbol)

                        for sf in symbol_first:
                            if sf in terminals:
                                first.add(sf)

                        if '^' not in symbol_first:
                            break
                    else:
                        break

                    if i + 1 == len(prod):
                        first.add('^')

        first_list.remove(X)

        #return first
        return first
```

if Grammar_dic value is '^' ,add '^' in first set for head. if not, that Grammar_dic value split!

〈Method Follow〉

```
def FOLLOW(A):
    follow_list = []

    #define set
    follow = set([])
    follow_list.append(A)

    #starting symbol add $
    if A == starting_symbol:
        follow.add('$')
```

compute A of Follow set.

if A is starting symbol, add "$" in follow set.

```
    for (heads, prods) in Grammar_dic.items():
        for prod in prods:
            prod = prod.split()

            if A in prod[:-1]:
                first = First(prod[prod.index(A) + 1])
                follow |= (first - set('^'))

                if '^' in first and heads not in follow_list:
                    follow |= FOLLOW(heads)

            elif A in prod[-1]:
                if heads not in follow_list:
                    follow |= FOLLOW(heads)

    follow_list.remove(A)

    return follow
```

for loop in Grammar_dic.items. All of grammar split.

Where A in prod, If there is '^' (not in follow_list), follow U Follow(heads).

```
Grammar_dic={"CODE'": ['CODE'], 'CODE': ['VDECL CODE', 'FDECL CODE', 'FDECL', 'VDECL'], 'VDECL': ['vtype identifier semi
Grammar= {1: 'CODE -> VDECL CODE', 2: 'CODE -> FDECL CODE', 3: 'CODE -> FDECL', 4: 'CODE -> VDECL', 5: 'VDECL -> vtype id
starting_symbol="CODE'"
terminals=['identifier', 'lparen', 'comma', 'addsub', 'return', 'while', 'rparen', 'rbrace', 'multdiv', 'integer', 'lbrac
nonterminals=['CODE', 'FDECL', 'STMT', 'RHS', 'FACTOR', 'VDECL', 'ARG', 'RETURN', 'COND', 'EXPR', 'TERM', 'MOREARGS', 'BL
parsing_table = {0: {'comma': '', 'addsub': '', 'identifier': '', 'lparen': '', 'comparison': '', 'literal': '', 'integer
```

Grammar terminals, nonterminals, parsing_table which I have already computed by hands.

```
#file input(last term project output)
file_pj1= open('allG_test.out','r')
#file ouput(if the grammar is accepted or not)
fileout= open('allG_test_pj2.out','w')
#file input uses as a string.
output_of_project1=file_pj1.read()
output_of_project1=output_of_project1.lower()
buffer = (output_of_project1 + " $").split()
```

Input is output from last term project 1.

Ouput is from term project 2.

```
while True:
    try:
        s = int(stack[-1])
        step += 1

        #If there is no Symbols in parsing table, break
        if a not in parsing_table[s].keys():
            print("ERROR: Unrecognized Symbol", a)
            break
```

Syntax analyzer using method of first, follow. input, output, Grammar, terminals, nonterminals, parsing_table.starting symbol.
Set stack. If there is no symbol in parsing_table. break while loop.

```
        #If all grammar accepted, break
        elif parsing_table[s][a] == "acc":
            a = "GRAMMAR : ACCEPTED"
            fileout.write(a)
            print(a)
            break
```

if all grammar is accepted, make 'Accepted' msg and break while loop.

```
    #catch the TypeError and make some comment why the Error occured in output file.
    except TypeError:
        f = "GRAMMAR : REJECTED ₩n"
        g = "STACK: " + str(stack) + "<- Error occured from here."
        h = f + g
        print(f)
        print(g)
        print("₩nType Error!!")
        fileout.write(h)
        fileout.write("₩nType Error!!")
        break


    #catch the IndexError and make some comment why the Error occured in output file.
    except IndexError:
        f = "GRAMMAR : REJECTED ₩n"
        g = "STACK: " + str(stack) + "<- Error occured from here."
        h = f + g
        print(f)
        print(g)
        print("₩nIndex Error!!")
        fileout.write(h)
        fileout.write("₩nIndex Error!!")
        break
```

If there is an error, write in output file why the error occurs.