Programming Language

Assignment #3

- Copy 시 소스 제공자와 카피 당사자 공히 **해당 과제 전체 0점 처리.**
- 반드시 자기 자신의 생각과 글로 작성할 것. 다른 곳에서 자료 인용 시 출처 반드시 기재 **(미기재 시 50% 감점)**
- 한글/영문 상관 없음.

1. Answer the question.

Syntax error and semantic error are two types of compilation error.
Explain the difference between the two in a program with examples.

구문 오류(Syntax Error)란 프로그래밍 언어의 문법(Syntax)을 제대로 기술하지 않아 발생하는 오류이다. 식별자를 잘못 작성하거나, 구두점 또는 세미콜론을 빠트리거나, 괄호 쌍을 제대로 닫지 않은 경우를 말한다. 컴파일 언어의 경우 구문 오류은 컴파일시에 감지될 수 있지만 인터프리터 언어의 경우 프로그램 실행 중에 감지될 수 있다. 다음과 같은 예시를 구문 오류로 들 수 있다.

```
$inlcude <stdio.h;

// C 언어: include 전처리기 구문은 #include로 시작하고, include할 대상 헤더 파일은 "" 또는 <>로 감싸져 있어야 함 (#include <stdio.h>가 올바른 사용임)
```

```
System.out.println["Hello World!"];

// Java: System.out PrintStream 객체에는 println라는 이름의 함수만 존재함.

// System.out.println("Hello World");로 함수 호출을 하지 않고 배열 색인 연산자를 사용했음.
```

의미 오류(Semantic Error)란 프로그래밍 언어의 문법에는 문제가 없지만, 프로그램의 실행 결과가 의도한대로 나오지 않는 오류를 말한다. 프로그래밍 언어 문법 상의 오류가 아니기 때문에 컴파일러 또는 인터프리터가 오류 메시지를 출력하지 않고도 발생할 수 있다. 다음과 같은 예시를 의미 오류로 들 수 있다.

```
# N의 네제곱을 구하기

N = 7

print (N * 4)

# Python : 곱 연산자(*)가 아니라 제곱 연산자(**)를 사용하여야 함 "print(N ** 4)"가 올바른 표현임
```

```
# N의 네제곱을 구하기

#include <stdio.h>

#include <stdbool.h>

int main() {

    bool is_full = true;

    if(~is_full){

        printf("Not Full");

    }

}
```

C99 : 올바른 부정 논리 연산자(!)가 사용되지 않음, 비트 부정 연산 연산자를 사용하였음 (bool은 1 바이트 부호있는 정수 자료형이고, true(DEC: 1, BIN: 00000001) 값이 할당된 후 비트 부정 연산 되어 DEC: -2, BIN: 11111110이 되고 C 언어에서 조건(if) 구문은 구문 내부의 평가 값이 0이 아닐 때 실행되므로 "Not Full"이 출력됨)

2. Answer the question.

(a)

Write a BNF description of the relational operator of Java, including the two operators == and !=.

<operator> → < | <= | > | >= | == | !=

Expr 사용) <expr> → <expr> <operator> <expr> | <expr>

(b)

Write EBNF descriptions for the following:

d. A C **union** definition
e. C **float** literals

d. A C union definition

<C union> → union <union_name> {

{<type> <var>;}+

}

<C union> → union <union_name> {

{<type> <var>;}+

} <var>;

<C union> → typedef union <union_name> {

{<type> <var>;}+

} <alias>;

*. <type> (char, short, int, long, long long, float, double, union, struct, pointer type 을 표현한 EBNF 표현식)과 <var>, <union_name>등 은 C99 문법 규칙을 준수하여 정의된 표현식이라 하자.

e. C float literal

<float> → (+ | -) { <integer> }+ . { <integer> }+ | (F, f)

<integer> → { ('0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9') }+

3. Answer the question.

Consider the following grammar:

<S> → <A> a <B> b

<A> → <A> b | b

<B> → a <B> | a

Which of the following sentences are in the language generated by this grammar?

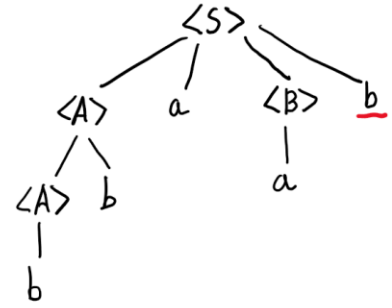
a. bbaabb

b. bbaba

c. bbaaaabb

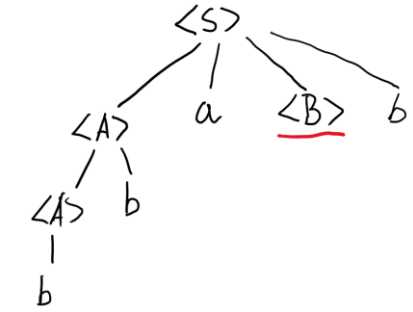
d. abaabb

a. bbaabb



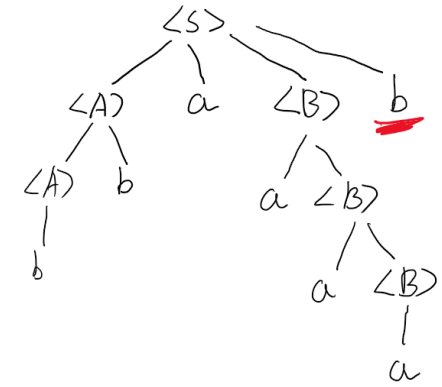
bbaa**b** 이후 b가 더 이상 존재할 수 없기 때문에 문법에 의한 생성이 불가능하다.

b. bbaba



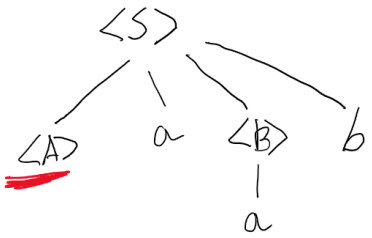
bba**<B>**b의 <B> 부분에서 a가 나오지 못하므로 문법에 의한 생성이 불가능하다.

c. bbaaaaaa



bbaaaaa**b** 에서 마지막 b 노드(빨간색 밑줄) 없이 구성이 불가능하므로 문법에 의한 생성이 불가능하다.

d. abaabb



<A>aab의 <A>에서 a가 나올 수 없으므로 문법에 의한 생성이 불가능하다.

4. Answer the question.

Convert the following EBNF to BNF:

S → A{bA}

A → a[b]A

<S> → <S> b <A> | <A>

<A> → a <A> | a b <A>

<The End of the Assignment>