Exercise 1 (Circularity of attribute grammars): (6 Points)

Consider the following grammar $G = (N, \Sigma, P, S)$ with inherited attributes $i1, i2$ and synthesised attributes $s1, s2$.

$$\begin{array}{lcll}
S' & \to & S & i1.0 = 1 \\
 & & & i2.0 = 2 \\
 & & & i1.1 = s1.1 \\
 & & & i2.1 = i1.0 \\
 & & & s2.0 = s2.1 \\
S & \to & AA & i1.1 = s1.1 \\
 & & & i2.1 = i1.0 \\
 & & & i1.2 = 0 \\
 & & & i2.2 = i2.0 \\
 & & & s1.0 = s2.1 \\
 & & & s2.0 = s2.2 \\
S & \to & A & i1.1 = 0 \\
 & & & i2.1 = i2.0 \\
 & & & s2.0 = s2.1 \\
A & \to & a & s1.0 = 0 \\
 & & & s2.0 = i1.0 \\
A & \to & b & s2.0 = 0 \\
 & & & s1.0 = i2.0
\end{array}$$

a) Provide the dependency graph for each production in G.

b) Apply the circularity test from the lecture to G.

1. Calculate the set $IS(A)$ for all $A \in N$.
2. Is G circular? Justify your answer.

c) To simplify the circularity test we want to consider so-called strong circularity. To this aim we modify our circularity test in a way, that attribute dependencies caused by different syntax trees are not distinguished anymore. Thus $IS(A)$, $A \in N$, is now defined as follows:

$$IS(A) = \{(\beta, \alpha) | \beta \overset{A}{\hookrightarrow} \alpha \text{ in some syntax tree t with root label A}\} \subseteq Inh \times Syn$$

with $\beta \in inh(A), \alpha \in syn(A)$.

Hint: $IS(A)$ is not a system of attribute dependence sets anymore, but a union!

If we adapt the circularity test from the lecture to this definition of $IS(A)$, does it provide the result that G is strongly non-circular? Argue why!

Exercise 2 (Jasmin): (4 Points)

Before we turn our attention to code generation, we need to understand our target language, the language of Jasmin.

> "Jasmin is an assembler for the Java Virtual Machine. It takes ASCII descriptions of Java classes, written in a simple assembler-like syntax using the Java Virtual Machine instruction set. It converts them into binary Java class files, suitable for loading by a Java runtime system."

Resources:

- Download Jasmin (`http://jasmin.sourceforge.net/`)

- Make sure you are able to execute it, try to compile and execute the example files
- For further information consult the Jasmin guide (`http://jasmin.sourceforge.net/guide.html`).

Hints:

- If you wonder what the `I` and `L` in `invokestatic java/lang/String/valueOf(I)Ljava/lang/String;` stand for, check out `http://www.cs.sjsu.edu/~pearce/modules/lectures/co/jvm/jasmin/data.html`
- In Jasmin comments are started with a semicolon, however sometimes the semicolon is part of a method call so mind the difference between:
  - `bipush 10 ;cmt` *(the space before the comment ";cmt" is mandatory!)*
  - `invokestatic java/lang/String/valueOf(I)Ljava/lang/String;` *(the semicolon is part of the statement and must follow it without spaces!)*

There is no framework for this exercise.
Tasks:

1. Write a Jasmin program that prints "Hello World" to the command line.
2. Write a Jasmin program that reads an integer from the command line, increases it by one and prints the result on the command line (no error handling required).
3. In Jasmin, implement a loop that increases a counter from 0 to 10. On every iteration, if the current value is odd, print the value on the command line.
4. Write a Jasmin program that evaluates $3 + (5 * 2)$ and prints the result on the command line.

Write a single file for each subtask, hand in all files packed in one zip file via L2P.