The C programming language does not feature type inference: types are rather informed by the developer through annotations, e.g.: `int x` will inform the compiler that variable x must have type int. There would be many challenges to add type inference to C. In this exercise, we shall see some of these challenges.

(a)

```
Missing declarations

int main() {
  a * b;
  a + c;
}
```

(b)

```
Missing declarations

int main() {
  a * b;
  a c;
}
```

(c)

```
Missing declarations

int main() {
  a * b;
}
```

Figure 1: (a-b) Two C programs where variable `a` belongs to different syntactic categories. (c) A C program where `a` is an ambiguous symbol.

UFMG
DCC
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
Compilers Laboratory

1. How would you have to complete the missing declarations in Figure 1 (a) to ensure that the program in that figure compiles?

2. How would you have to complete the missing declarations in Figure 1 (b), to ensure that the program in that part of the figure also compiles?

3. What about the program in Figure 1 (c): is it possible to discover what is the syntactical category of symbol `a`?

4. Can you think about a general algorithm to infer the syntactical category of symbols like variable `a` in the three code snippets above?

5. Try to explain how your algorithm works. Does it pass over the program one single time? Does it generate constraints? Does it work in multiple phases? Does it involve a lattice (you might want a lattice to distinguish between pointer types and numeric types, and separate types like int, double, char, etc…)?

To know more: Leandro T. C. Melo, Rodrigo Geraldo Ribeiro, Breno Campos Ferreira Guimarães, Fernando Magno Quintão Pereira: *Type Inference for C: Applications to the Static Analysis of Incomplete Programs*. ACM TOPLAS 42(3): 15:1-15:71 (2020)