This question refers to the program below, which is implemented in Python. Python is a programming language that represents objects pretty much as dictionaries. Yet, there are more efficient representations!

```
01 def test(i):
02     v = OX()
03     if i % 2:
04         tmp = i + 1
05         v.m1(tmp)
06     else:
07         v = OY()
08         v.m2()
09     print v.m3()
```

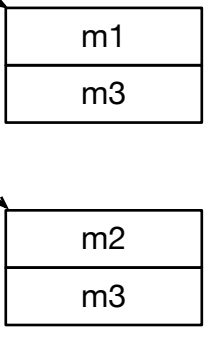

Figure 1: A simple python program that invokes methods onto object v.

1. Have you ever heard about virtual tables? What are they?

2. Can you convince yourself that these drawings show the right virtual tables for classes OX and OY?

UFMG
DCC
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
Compilers Laboratory

3. Would it make sense to design an analysis to add virtual tables to Python objects? What would be the difficulties of getting such an analysis right?

4. Can you design a static analysis that finds a conservative approximation for the virtual tables of classes in Python?

5. Would it be possible to implement this analysis using linear space in terms of program size? I mean, if you have N instructions in (some) low level representations of the program, then your analysis should run in time proportional to O(N) and should use memory proportional to O(N). What do you say?

6. How does information propagate in your analysis? Where does it appear?