CS4215 Programming Language Implementation

Lab task for Week 06
A Garbage Collector for simPL

1. Download the file `http://www.comp.nus.edu.sg/~cs4215/labtasks/week6.zip`, and extract it to the Eclipse workspace folder. The workspace folder should now contain a file `cs4215_week_6`.

2. In Eclipse, go to "File", "New", "Project", "Java Project", "Next", and choose "`cs4215_week_6`" as "Project name". Press "Finish".

3. Use the "Run Configurations" to run `simPLcompiler.simplc` with a file name (for example `test.simpl`) as "Program argument". The file can contain any simPL program. The compiler should reply:

   ```
   sVML code written to test.svml
   ```

   Now, you can interpret the compiled program using the virtual machine by running `simPLvm.simpl` with the base name of the file you just compiled (in the example `test`.

   Note that the given virtual machine in `VM.java` uses the ideas outlined in Section 9.4. When the machine runs out of memory—see function `New` in `VM.java`—it simply reports "Memory Exhausted" and the machine crashes with an array-out-of-bounds exception. It is your task to extend this virtual machine to implement Cheney's copying garbage collector.

   We will test the efficacy and correctness of the garbage collector by running tests that push the size limits of the heap. Therefore it is important that you do not try to change `HEAP_SIZE`; as the size of your heap, you need to use the `HEAP_SIZE` that is inherited from `FixedSizeVM`.

   Hints:

   - You may stick with the current setup, where the runtime stack lies outside the heap, and each stackframe is allocated on the heap. Therefore, the runtime stack simply contains integers that represent the addresses of the stackframes on the heap. In this setup, what are the roots of the garbage collector?
   - Note that the tags that identify the type of each node are all negative. One idea is therefore to use this tag slot as forwarding address, since real addresses are non-negative, and thus cannot be confused with tags. This way, you would not have to change the node layout at all.

- Be careful with mutator calls. Which instructions that have several mutator calls? Do you see a problem when garbage collection happens during the execution of these instructions?
- Find more hints and discussions in the forum. Let us try to share benchmark simPL programs, as well.

Submit the resulting file

- `VM.java`

from your folder `simPL` in the IVLE workbook "Week 6".

Make sure that you do not change any other files when you test your programs.

Suggestion: When you are done with the solution, save your four files in a secure place. Then download a fresh copy of the lab task, and place your three files into that copy. Then re-do your tests.