Practical #7:
Liveness Analysis

COS**341** Compiler Construction

Universiteit van Pretoria

Academic Year 2015

Submission due: Wednesday the **6th of May**, mid-day (online)
Presentation due: Wednesday the **6th of May**, evening (laboratory)

The usual **terms and conditions** [see Study-Guide] are **applicable without exception**

INPUT

- Once again the Tutor in the Lab will provide you with a **BASIC Program**, which we will use as our "*Intermediate Code*".

Task A) ..... [½ Mark]

- For each instruction ***i*** in the given input program, **compute** its successor set

***succ*[*i*]**,

as per **Section 8.2** of our Textbook.

- *For the implementation you may use any programming language of your choice, as long as it "does the job".*

Task B) ..... [¾ Mark]

- For each instruction ***i*** in the given input program, **compute** its "generator" set

***gen*[*i*]**,

as per **Section 8.2** of our Textbook,

(particularly *Table 8.1* on page **162**).

- *For the implementation you may use any programming language of your choice, as long as it "does the job".*

Task C) ..... [¾ Mark]

- For each instruction ***i*** in the given input program, **compute** its "killer" set ***kill*[*i*]**,

  as per **Section 8.2** of our Textbook,
  (particularly *Table 8.1* on page **162**).

  - *For the implementation you may use any programming language of your choice, as long as it "does the job".*

Task D) ..... [**1** Mark]

- For each instruction ***i*** in the given input program, **compute** its "out-liveness" set

***out*[*i*]**,

as per **Section 8.2** of our Textbook,

(particularly *Equation 8.2* on page **161**).

- *For the implementation you may use any programming language of your choice, as long as it "does the job".*

Task E) ..... [**1** Mark]

- For each instruction ***i*** in the given input program, **compute** its "in-liveness" set

$$in[i],$$

as per **Section 8.2** of our Textbook,

(particularly *Equation 8.1* on page **161**).

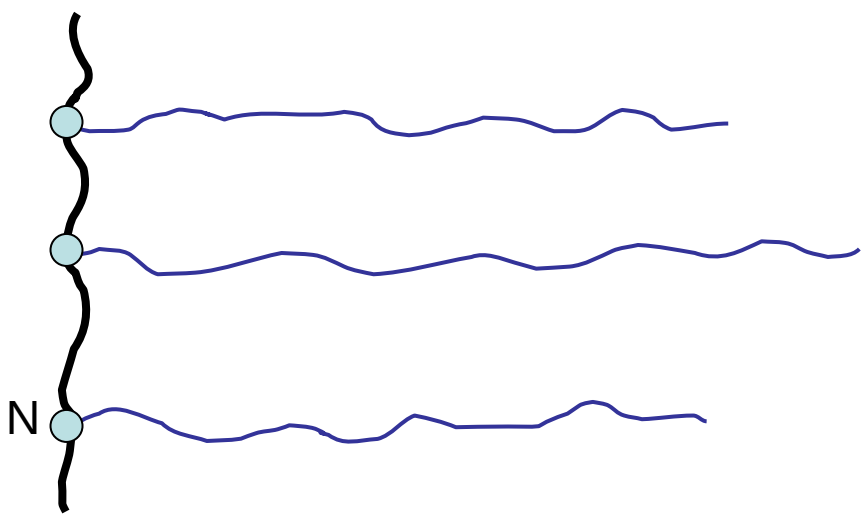
- *For the implementation you may use any programming language of your choice, as long as it "does the job".*

Professor's "**Hints and Advice**" for a "**smart implementation**"

- Remember the facts that
  - all the succ[*i*] = **{...}**, gen[*i*] = **{...}**, kill[*i*] = **{...}**, out[*i*] = **{...}**, in[*i*] = **{...}** are *mathematical* ***sets***
  - the input BASIC program has *line numbers* **N**
- These observations motivate the utilisation of *built-in hashtables* for the representation of those sets,
  - *whereby the line numbers* **N** *might be used as "**keys**" to the "buckets" of those hashtables.*

Professor's "**Hints and Advice**" for a "**smart implementation**"

- In case that you do not feel comfortable with hashtable implementations, you can also use other linked data-structures that can be accessed via the line-numbers N as entry points.



Professor's "**Hints and Advice**" for a "**smart implementation**"

- Moreover, please keep in mind that the functions (methods, procedures) for the computations of the **in**[...] and **out**[...] sets must call each other in **Mutual Recursion** (until the fixpoint has been reached when nothing changes any more):
  - *For Mutual Recursion please revisit your old Algorithms & Data Structures Textbook from* ***COS212***

And now :

HAPPY CODING