SNU 4541.664A Program Analysis, Spring 2005
Homework 2

due: 4/09 24:00 (submit your program and report to `kwang@ropas`)

For the following two homeworks, you can team with others. Team size ≤ 2.

Exercise 1 "Interpreter"

Consider the following imperative language C--:

program	pgm	$\rightarrow$	c
command	c	$\rightarrow$	x `:=` $e \mid c$ `;` c
		$\mid$	`if` e `then` c `else` c
		$\mid$	`while` e `do` c `end`
		$\mid$	`local` x `:=` e `in` c `end`
expression	e	$\rightarrow$	`readint` $\mid z \mid x$
		$\mid$	e `+` $e \mid e$ `-` $e \mid e$ `*` e
		$\mid$	e `<` $e \mid e$ `=` e

Command changes the memory. Expression computes a value. Command assigns a value to a memory location denoted by a variable, does a sequence of commands, branches based on condition's zero-ness, repeats the while-body while condition value is non zero, limits a scope of a variable to its local block. Expression reads an integer from the outside world, is a constant integer, is the value of a variable, or is one of the usual integer or boolean operations.

Implement its interpreter `eval`

$$\texttt{eval} : \mathit{Memory} \times \mathit{Program} \rightarrow \mathit{Memory}$$

in your favorite language.

Those who is not religious about your favorite languages, I recommend you to use nML(`ropas.snu.ac.kr/n`). As an encouragement, I will provide the C-- parser in nML in the class homepage. □

Exercise 2 "Just try"

The C-- will be used to program the inertia navigation system of Korean liquid-fuel rocket KSR-XII. C-- programs will control the KSR-XII rocket until it reaches its orbit.

Because KSR-XII's engineers who experienced many failures of the predecessor rockets and found all the failures were due to some software bugs, this time they want to make sure that their software is bug-free.

Your company offered them to provide a software technology for the problem: static analysis. KSR-XII's definition of bug-freeness is: every integer variable must have values within particular ranges. For example, some variable that determines the rocket's throttle valve must not exceeds some limit. Hence, the static analysis must be able to estimate all the values of every variables in their C-- programs.

Because the C-- variables can only have integer values, one way to estimate a set of integer values is to compute the interval (the minimum and maximum) of the integer values. For example, consider the following program. The numbers between commands are indices for program points.

```
1:
   x := 1;
2:
   while x < 1000 do
3:
      x := x+1
4:
   end
5:
```

You may hope that your analysis concludes that for the program points the integer intervals of `x` are:

1: *undefined*
2: $[1, 1]$
3: $[1, 999]$
4: $[2, 1000]$
5: $[1000, 1000]$

As another example, consider the following program.

```
1:
   x := 1;
2:
   y := 2;
3:
   if x
4:                        5:
   then z := x + y                else w := y * -1
6:                        7:
8: (* after the two branches *)
```

You may hope that your analysis concludes that for each program point the

integer interval of the variables are:

1: *undefined*
2: $\{\mathtt{x} \mapsto [1,1]\}$
3: $\{\mathtt{x} \mapsto [1,1], \mathtt{y} \mapsto [2,2]\}$
4: $\{\mathtt{x} \mapsto [1,1], \mathtt{y} \mapsto [2,2]\}$
5: $\{\mathtt{x} \mapsto [0,0], \mathtt{y} \mapsto [2,2]\}$
6: $\{\mathtt{x} \mapsto [1,1], \mathtt{y} \mapsto [2,2], \mathtt{z} \mapsto [3,3]\}$
7: $\{\mathtt{x} \mapsto [0,0], \mathtt{y} \mapsto [2,2], \mathtt{w} \mapsto [-2,-2]\}$
8: $\{\mathtt{x} \mapsto [0,1], \mathtt{y} \mapsto [2,2], \mathtt{w} \mapsto [-2,-2], \mathtt{z} \mapsto [3,3]\}$

Design, implement, and see how good is your analyzer. You may want to play with the `eval` when you implement your analyzer.

What would you say about a correctness of your analyzer? Can you also guarantee that your analyzer always terminate even for non-terminating programs like those used in KSR-XII?

Write a report on how you designed it and what, if any, problems you experienced. We will discuss each of your solutions in class. □