Formal Methods in Software Development
Exercise 4 (June 1)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.uni-linz.ac.at

April 25, 2006

The result is to me submitted to me by **June 1** (hard deadline) as an email containing a zipped file (.zip or .tgz) with the JML-annotated Java file, all files generated by Krakatoa/Why, and the .prf file of your PVS proof of the generated verification conditions.

1 Finding a Maximum (Java Verification)

Verify with Krakatoa/Why and PVS the total correctness of the Java method

```
class Main
{
  public static int findMax(int[] a)
  {
    int m = -1;
    int n = a.length;
    for (int i=0; i<n; i++)
    {
      if (m < a[i]) m = a[i];
    }
    return m;
  }
}
```

for finding the maximum of an array of integers.

For fulfilling this task, you have to perform the following steps:

1. Write an appropriate JML header specification for `findMax` (type-checking it with `jml`). You should note that the method only works under certain assumptions on the size and the contents of the array.

   Before proceeding, validate your specification with `escjava2` (which should not give any warnings).

2. Write an appropriate JML loop invariant (`loop_invariant`) and termination term (`decreases`) for the loop (type-checking them with `jml`). The `for` loop can be annotated like a corresponding while loop making use of the loop variable `i`.

3. Run `krakatoa -pvs Main_findMax` and `make pvs` to generate the file `Main_findMax_why.pvs` containing the proof obligations.

4. Prove the generated obligations with PVS.

   (a) Of the six generated obligations, three can be verified by a single application of `grind`.

   (b) The other three obligations can be verified by application of the proof commands `skosimp*`, `skolem`, `flatten`, `assert`, `split`, `inst`, and of course `grind`.

   None of these proofs is difficult; you more or less have to apply above commands (already listed in the right order ;-) to simplify your proof situations. In two of the proofs, you have to apply `skolem!` to an existential assumption in order to generate a constant with which an universal goal can be instantiated by application of `inst`.

Even if you are not successful with all proofs, proceed as far as possible and submit to me your partial results.