Homework 6

Please submit your solution via email to the instructor with CC to ly603@nyu.edu.

The deadline for Homework 6 is March 25.

Problem 1 JML Tools (3 Points)

Download and install JML Tools from:
`http://sourceforge.net/projects/jmlspecs/files/`
Note that you should use the *jmlspecs* distribution, not the *OpenJML* distribution, as *OpenJML* does not yet support all the JML features needed for this homework assignment. Get yourself familiar with the JML compiler (`jmlc`) and runtime assertion checker (`jmlrac`).

Problem 2 Heavyweight Specifications (6 Points)

Consider the following Java method:

```
static int f(int n){
  int i = 0;
  int s = 0;
  while (s < n) {
    i = i + 1;
    s = s + 2 * i - 1;
  }
  return i;
}
```

Write a heavyweight JML specification for method `f` that precisely characterizes the method's return value for non-negative input values `n`. You may assume that the Java type `int` can hold arbitrary large integer values.
Test your specification with the JML runtime assertion checker (`jmlrac`). You will need a `main` method that calls `f` for various input values. *Remember:* You have to compile your annotated Java code with the JML compiler (`jmlc`) before you can use the runtime assertion checker.
What happens if you test your specification for input values greater than 2147395600? Why this value?

Problem 3 Map Implementation (16 Points + 4 Bonus Points)

On the course webpage you find the skeleton for a Map data structure that implements functions mapping keys to values using sorted binary trees. Your task is to implement and specify this data structure. The specification should be given in terms of a model field of type `JMLValueToObjectMap`, so make yourself familiar with the methods provided by this class.

(a) Implement and specify a method

```
private /*@pure@*/ JMLValueToObjectMap computeContent(Node n)
```

that computes the content of the binary tree. **(4 Points)**

(b) Implement and specify a method

```
public /*@pure@*/ boolean inDomain (Key k)
```

that checks whether there exists a node in the tree whose key is equal to `k`. **(4 Points)**

(c) Implement and specify a method

```
public /*@nullable@*/ Object add(Key k, Object v)
```

that inserts a key/value pair into the tree. The tree should remain sorted. If a node with that key already exists, the old mapping will be replaced. The methods returns `null` if the key was not mapped to a value before this method has been called, and the old value otherwise. **(4 Points)**

(d) Write a small test program that creates an instance of your map and inserts some key/value pairs where keys are of type `IntKey`. Compile and run your program with the JML tools. **(4 Points)**

(e) **Bonus:** Give a class invariant that states sortedness of the tree. Test your invariant with your example program. *Hint: There are multiple ways to do this, but all require you to write at least one auxiliary pure method.* **(4 Bonus Points)**