

## Fall 2008 Software Engineering Ph.D. Qualifying Exam

Below is some annotated Java code that will be used for several problems. The ST class implements a symbol table; the STObject interface is necessary interface for things stored in an ST collection, and the STInt class is an example class that can be stored in an ST collection.

```

001 interface STObject
002 {
003     public int Key();
004 }
005
006 class STInt implements STObject
007 {
008     public int d;
009     public int r;
010     public int Key()
011     {
012         return r;
013     }
014 }
015
016 class ST
017 {
018     private /*@ spec_public @*/ STObject table[];
019     private /*@ spec_public @*/ int numentries;
020     private /*@ spec_public @*/ int lastemptyentry;
021
022     // after constructor, assert that we will have
023     // a non-null, non-empty table
024     // -----
025     /*@ invariant table != null;
026     /*@ invariant table.length > 0;
027
028
029     /*@ requires size > 0;
030     /*@ ensures numentries == 0 &&
031         lastemptyentry == -1;
032     public ST(int size)
033     {
034         table = new STObject[size];
035         numentries = 0;
036         lastemptyentry = -1;
037
038     /*@ ensures \result >= 0 &&
039         \result < table.length;
040     private int Hash(int key)
041     {
042         int t = key % table.length;
043         if (t < 0)
044             t = -t;
045         return t;
046     }
047     /*@ ensures \result >= 0 ==>
048         \result < table.length;
049     /*@ ensures \result < 0 ==>
050         (lastemptyentry >= 0 &&
051         lastemptyentry < table.length); @*/
052     private int Find(int key)
053     {
054         int i = Hash(key);
055         while (i < table.length)
056         {
057             if (table[i] == null)
058             {
059                 lastemptyentry = i;
060                 return -1;
061             }
062             else if (table[i].Key() == key)
063             {
064                 return i;
065             }
066             i++;
067         }
068         return -1;
069     }
070     /*@ requires sto != null;
071     public int Add(STObject sto)
072     {
073         int i;
074         i = Find(sto.Key());
075         if (i >= 0)
076             return 0;
077         table[lastemptyentry] = sto;
078         numentries++;
079         lastemptyentry = -1;
080         return 1;
081     }
082
083     public /*@ nullable @*/
084     STObject Get(int key)
085     {
086         int i = Find(key);
087         if (i >= 0)
088             return table[i];
089         else
090             return null;
091     }
092     /*@ requires args != null;
093     public static void main(String args[])
094     {
095         int i, stat;
096         STInt iobj;
097         ST cache =
098             new ST(Integer.parseInt(args[0]));
099         for (i=1; i < args.length; i++)
100         {
101             iobj = new STInt();
102             iobj.r = i;
103             iobj.d = Integer.parseInt(args[i]);
104             stat = cache.Add(iobj);
105             System.out.println(
106                 "added object: (" + iobj.d +
107                 ", " + iobj.r + ") " + stat);
108         }
109         while (i >= 1)
110         {
111             iobj = (STInt) cache.Get(i);
112             if (iobj != null)
113             {
114                 System.out.println(
115                     "found object: (" +
116                     iobj.d + ", " + iobj.r + ")");
117             }
118             else
119             {
120                 System.out.println(
121                     "no object with key: " + i);
122             }
123             i--;
124         }
125     }

```

---

This page intentionally left blank.

---

## [20pts] 1. UML

[10pts] A) Draw a UML class diagram for this program, being sure to provide proper labeling as needed, and having direction, cardinality, and modality on associations that need it.

[10pts] B) Draw a UML message sequence diagram for all messages that occur from an invocation of the `ST.Add` method, assuming that the object being added is not in the table already, but that another object exists in the table at the same initial position that the new object's key hashes to. There are no other objects in the table.

---

## [20pts] 2. JML

[5pts] A) What do the JML tools "jmlc" and "jmlrac" do with the annotations on the program?

[7pts] B) The annotations on the above program are fairly simple, and only add a constant overhead to the running program (i.e., if the program is  $O(n)$ , it is still  $O(n)$  with the annotations being checked. However, it is possible to construct annotations in JML that can change the program from, say,  $O(n)$  to  $O(n^2)$  or even worse. Explain what types of annotations can do this, and describe how and why they do.

[8pts] C) The program above, when compiled and run using the JML tools, successfully runs and produces the following output:

```
> jmlrac ST 10 4 3 6 2
added object: (4,0) 1
added object: (3,1) 1
added object: (6,2) 1
added object: (2,3) 1
no object with key: 4
found object: (2,3)
found object: (6,2)
found object: (3,1)
found object: (4,0)
```

With the output above, can we conclude then that the program is correct in relation to the specifications in the annotations? Why or why not?

---

## [30pts] 3. ESC/Java

[5pts] A) ESC claims that it does *static modular* checking. Explain what this means.

[5pts] B) ESC's loop analysis is *unsound*. Explain why it is unsound, and why ESC chooses to analyze loops this way.

[5pts] C) ESC passes the "Hash" method above as OK, even if lines 42 and 43 are removed. Yet the modulus operator in Java will produce a negative result if the numerator is negative. Explain this apparent discrepancy.

[5pts] D) When ESC is analyzing each method in the `ST` class, what role do the *invariant* annotations play?

[10pts] E) The program above, when analyzed with the ESC/Java tool, passes except for two insignificant warnings (lines 77 and 108) and one other warning that looks significant, shown below:

```
> escj ST.java
.:.
ST: Find(int) ...
-----
ST.java:68: Warning: Postcondition possibly not established (Post)

Associated declaration is "ST.java", line 48, col 5:
/*@ ensures \result < 0 ==> (lastemptyentry >= 0 && ...
^
Execution trace information:
  Reached top of loop after 0 iterations in "ST.java", line 53, col 4.
  Executed else branch in "ST.java", line 60, col 12.
  Executed else branch in "ST.java", line 60, col 12.
  Reached top of loop after 1 iteration in "ST.java", line 53, col 4.
  Executed return in "ST.java", line 67, col 4.
...

```

Decide whether or not this is a significant warning. Explain how and why you reached this decision. If it is significant, what is the correct fix?

---

## [30pts] 4. Testing

For the parts below we are using the definitions and methods developed in the Harrold and Rothermel paper, "Performing Dataflow Testing on Classes".

[5pts] A) Write down all the *intra-method* def-use pairs for the uses of the variables *i* and *key* on line 60.

[5pts] B) Write down all the *inter-method* def-use pairs for the use of *lastemptyentry* on line 77.

[5pts] C) Write down all the *intra-class* def-use pairs for the use of *lastemptyentry* on line 77.

[5pts] D) The JML question demonstrated the program running with a command-line input of "10 4 3 6 2". For parts A, B, and C, which def-use pairs are exercised in this execution, and which are not?

[5pts] E) Create an execution (or more if you need them) that will exercise the def-use chains that were not exercised in part D. An execution is only specified by its command-line arguments.

[5pts] F) On line 111, the "r" field on an STInt object is used. What are the reaching definitions for this use? Why might this be extremely hard for an algorithm that finds def-use pairs to find?