Testing Project

CSE 4321, Spring 2016

Motivation

- Provides an opportunity to practice the basic concepts, principles, and techniques covered in this course
- In particular, you will apply control flow testing to a program of moderate size
  - RBTree.java: a Java class that implements a red black tree (available in Blackboard)
  - about 800 lines of code and with a number of seeded faults

Deliverables

- Final Report, Due by May 5th (midnight), 2016
  - Artifacts: CFG for each method, test paths for edge coverage, test data for each path, code coverage report, faults detected and corrections
  - Summary and discussion, lessons learned, and anything that you want to share

Recommendations

- NetBeans
- Code Coverage - JaCoCo

Deliverables

- Files:
  - 1. “CFG.pdf(35 %)” – For each method, provide Source Code, Block table, CFG.
  - 2. “TR for Edge Coverage.pdf(20 %)” – For each method, if possible, provide Test paths, actual input value for each of the corresponding test paths to cover all the edges. If not possible, indicate which one(s) cannot be covered.
  - 3. “Faults Detected and Corrections.pdf(7 %)” – For each method, provide the fault(s) and correction(s) to the fault(s). Show change of Source Code.
  - 4. “Code Coverage Report.pdf(1 %)”
    - Test over methods
    - End to end test with program inputs
  - 5. “Summary and Discussion.pdf(1 %)”
  - 6. Source Code(35 %)
  - 7. “Read me.txt(1 %)”

Block table

Example

```
1.  begin
2.     int x, y, power;
3.     float z;
4.     input (x, y);
5.     if (y < 0)
6.        power = -y;
7.     else
8.        power = y;
9.     z = 1;
10.    while (power != 0) {
11.       z = z * x;
12.       power = power – 1;
13.    }
14.    if (y < 0)
15.       z = 1/z;
16.    output (z);
17. end;
```

Block	Lines	Entry	Exit
1	2, 3, 4, 5	2	5
2	6	6	6
3	8	8	8
4	9	9	9
5	10	10	10
6	11, 12	11	12
7	14	14	14
8	15	15	15
9	16	16	16

Evaluation

- Documentation :
  - Well-structured, well-written, easy-to-read
- Technical Correctness:
  - All the artifacts are computed correctly
  - Try to detect as many faults as possible

Reference for Red Black Tree

- Chapter 13 in Introduction to Algorithms written by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein.
- http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-introduction-to-algorithms-sma-5503-fall-2005/video-lectures/lecture-10-red-black-trees-rotations-insertions-deletions/
- http://en.wikipedia.org/wiki/Red%E2%80%93black_tree