**Question 1**

1. How many classes has the group implemented?

2. Are class names nouns? Do the chosen class names make sense? Are they intuitive?

3. Does each class have an author?

4. Are method names verbs? Do the chosen method names make sense? Are they intuitive?

5. Does the implementation contain any class hierarchies?

6. Do classes in the hierarchies appear to reflect the is-kind-of relationship? If not, which ones don’t seem to?

7. Do classes in the subsystems appear to reflect the is-part-of relationships? If not, which ones don’t seem to?

8. Does the software design make sense? Have decisions been justified?

How well is the software designed?

• bad fail (< 20): design guidelines are poorly followed

• fail (20–40) • 3rd (40–50) • 2ii (50–60)

• 2i (60–70) • 1st (70–80)

• good first (> 80): software design is flawless.

**Question 2**

1. Has the group provided screen capture movies that demonstrate the features of their software?

2. Does the source code compile and provide a running application?

3. For each group member, how many classes did they implement?

4. How closely does the source code conform to the coding conventions? [2]

5. List the classes that violate one of Bob’s Concise Coding Conventions (C 3 ) [2]. The most commonly neglected rule is #10-use symbolic constants rather than raw numbers when writing source code. One point is deducted each time a rule is broken.

6. Are different sizes of game boards supported?

7. Can each game be played to completion?

8. Is there a start new game option?

9. Is the name of the player shown?

10. Is the elapsed time shown?

11. Are the number of diffused mines, total number of mines, and total number of both hidden and revealed squares shown?

12. Is diffusing squares working properly?

13. Are two types of players supported?

• person

• computer

14. Is the reveal mines feature working?

15. Are there animations for the game victory and player moves?

16. Are player pieces animated?

17. Can users save and re-load games?

18. Is game winner indicated in a fun and cool way indicating the winning move?

19. Can a game be saved and re-loaded?

20. Is the user informed if they attempt an invalid move?

21. Do the functions specified in the group report actually work as advertised?

22. Are there any important features missing with respect to the assignment specification?

23. How does the software rate in terms of usability? Is it intuitive? How well is the software implemented?

• bad fail (< 20): poor and incomplete functionality

• fail (20–40)

• 3rd (40–50)

• 2ii (50–60)

• 2i (60–70)

• 1st (70–80)

• good first (> 80): complete functionality with no bugs

**Question 3**

1. Have unit tests been defined for all of the classes? If not, for how many classes have unit tests been defined? (Hint: Do not comment your unit tests out.)

2. Try running the unit test for each class. Do they actually run and work? Note: It’s fine if unit tests were written that test more than one class at a time. They just have to be documented (noted in the doxygen comments).

3. Is the “Revealing Algorithm” working properly?

4. When does the application crash? Which features cause it to crash?

5. Can the application handle random input?

6. What happens when the input file is corrupted?

7. For example, what if you replace the normal user-data file with a JPEG file? 8. What happens if you edit the user-data file with a text editor and give it an unexpected format?

9. How does the program respond when the user or input data file is missing?

10. Do you think the group did a good job of testing their application overall? How well is the software implemented?

• bad fail (< 20): poor and incomplete testing

• fail (20–40)

• 3rd (40–50)

• 2ii (50–60)

• 2i (60–70)

• 1st (70–80)

• good first (> 80): very thorough and complete testing resulting in a robust piece of software.

**Question 4**

1. Has a user data file format description been provided somewhere, e.g., in the class that reads/writes the data?

2. How complete is the doxygen output? Are both the short (brief) and detailed class descriptions there as described in Bob’s Concise Introduction to Doxygen [1]?

3. Are both the class hierarchy and collaboration diagrams there?

4. Are all methods described?

5. Are all input parameters to methods documented?

6. How about return parameters?

7. List methods that have undocumented input and return parameters and the corresponding class to which they belong.

8. Has the group output their source code as part of their doxygen output? How well is the software documented?

• bad fail (< 20): poor and incomplete documentation

• fail (20–40)

• 3rd (40–50)

• 2ii (50–60)

• 2i (60–70)

• 1st (70–80) • good first (> 80): very thorough and complete documentation resulting in a very understandable API of the software.

**Question 5**

1. Did the group follow the minutes of meeting protocol?

2. Did the group meet often enough?

3. Have all the required files and documents been provided, e.g., design document, group report, minutes, source code, class files, doxygen web pages, and demo movies?

4. Have all the required files and documents been provided on time? If not, how late was the group in delivering the product?

5. Have all of the files been provided in the correct format, e.g., .PDF, .mpeg, etc., as described in the Assignment?

6. Do all of the folders and file names conform to the naming conventions described in the Assignment? List the files (and folders) that do not adhere to the guidelines (if any).

7. Did each group manager contribute to the group report? Does the group report answer the questions from the lecture on group work and team roles?

8. Did each group member contribute to the project? List the group members that contributed and those that did not. Evidence of group member contributions can be found in four ways: (1) by observing lecture attendance (2) by looking at the author of Java source code (.java) files, (3) by reading the minutes of meeting protocol at who attended the meetings and (4) by reading the Group Report.

9. Did the group report any problems? Did they also report how they resolved those problems?

10. How well do you think the group worked together? Did the active members of the group make up for the inactive members?

How well did the group work together?

• bad fail (< 20): little or no team work and collaboration

• fail (20–40)

• 3rd (40–50)

• 2ii (50–60)

• 2i (60–70)

• 1st (70–80)

• Good first (> 80): the group worked well together, met regularly, and team member helped.