Algorithms Data Structures and Efficiency

Coursework: Designing a Small Game with Algorithms and Data Structures

1. Objective

The goal of this coursework is to design and implement a small game that requires the use of algorithms and data structures. Students will write a report explaining their design choices, the algorithms and data structures used, and an analysis of the efficiency of their implementation.

2. Game Design

Title: "Treasure Hunt"

Game Description:

The player controls a character on a 2D grid-based map. The goal is to find three hidden treasures while avoiding obstacles. The map is randomly generated, and the treasure locations are unknown to the player. The player can move in four directions (up, down, left, right) and must navigate the map efficiently to find all the treasures.

Game Features:

a) Map Generation: The game map is a 2D grid with obstacles, open paths, and the treasures placed randomly. The size of the map is 20*20, and the number of obstacles is greater than 10.
b) Player Movement: The player can move in four directions. If the player hits a wall, the movement is blocked.
c) Treasure: The player finishes the game by finding all three treasures.
d) Pathfinding: The player can optionally use a "hint" button that displays the shortest path to the treasure.

Game Rule:

a) The initial score is 100, which means the startup capital.
b) When the player makes the choice of the next move by himself/herself, it will cost 1 coin and the related score will be reduced by 1.
c) When the player makes the choice of the next move by using 'hint' help, it will cost 3 coins and the related score will be reduced by 3.
d) When the player bumps into the wall, the related score will be reduced by 10.

3. Requirements for Students:

Implement the Game:

a) Use JAVA programming (JavaSE-23), and make your code work in PMB306

b) Implement the game logic, including map generation, player movement, and treasure placement.

Use Algorithms and Data Structures:

Choose appropriate algorithms and data structures for the following tasks:

- Map Representation: Decide how to represent the grid
- Pathfinding: Implement a pathfinding algorithm to find the shortest path to each treasure.
- Treasure Placement: Ensure the treasures are placed in a reachable location

4. Write a Report:

a) Explain the design of your game and the role of algorithms and data structures in its implementation.
b) Describe the algorithms and data structures you used and justify your choices.
c) Analyze the efficiency of your implementation in terms of time complexity and space complexity.
d) Discuss any trade-offs or optimizations you made.

5. Report Structure:

a) Introduction:
   - Briefly describe the game and its objectives.

b) Design and Implementation:
   - Explain how you represented the game map and other entities (e.g., player, treasure).
   - Describe the algorithms and data structures used for: e.g.:
     - ➢ Map generation
     - ➢ Pathfinding
     - ➢ Treasure placement

c) Efficiency Analysis:
   - Analyze the time complexity and space complexity of your algorithms
   - Discuss how your choices impact the performance of the game

d) Challenges and Trade-offs:
   - Discuss any challenges you faced during implementation.
   - Explain any trade-offs you made (e.g., simplicity vs. efficiency, space and time used).

e) Conclusion:
   - Summarize your work and reflect on what you learned about algorithms, data structures, and efficiency.

6. Grading Criteria:

a) Report (50%):

    - Clear explanation of design choices.
    - Justification of algorithms and data structures used.
    - Thorough analysis of efficiency.

b) Code Quality (50%):
    - Readable, well-structured, and commented code.
    - Efficient and optimized implementation.

7. Deliverables:

a) Source code for the game. (zip file)

b) A report following the structure outlined above (2000-3000 words).

c) A short video (2-3 minutes) demonstrating the game and its features (optional but recommended).

- ❖ Code and video in one zip file, if you have video.

Deadline:

The completed source code zip file, written report, or demonstration video should be uploaded to the Coursework Submission link on the COMP2069 Moodle page **by 16:00 on April 25**. All the deliverables should be named using your ID. You may submit as many times as you wish and the last submission will be used for marking. However, if you submit after the deadline, your last submission time will be considered for the late submission penalty. After the deadline has passed, if you have already submitted your exercise then you will not be able to submit again. If you have not already submitted then you will be allowed to submit once.

Late submissions: Late submissions will incur a penalty of 5% per day. If you have extenuating circumstances you should file them before the deadline.

Plagiarism

Copying code or report from other students, from previous students, from any other source, or soliciting code or report from online sources and submitting it as your own is plagiarism and will be penalized as such. FAILING TO ATTRIBUTE a source will result in a mark of zero – and can potentially result in failure of coursework, module or degree. All submissions are checked using both plagiarism detection software and manually for signs of cheating. If you have any doubts, then please ask. Hence, comments must adequately explain your code. We reserve the right to ask students to explain code and/or comment on their report in exceptional circumstances, to demonstrate understanding.