CPT204-2324 Coursework 3 Task Sheet

Overview

Coursework 3 (CW3) is the final coursework component of the course this semester. It contributes to **40% of your final marks**.

You will form **a team of two** with your friend and apply object-oriented principles and advanced data structures you have learned throughout the semester to create intelligent game-playing characters. You will be tasked with writing a report and creating a video presentation to demonstrate your problem-solving and testing skills, as well as your understanding of object-oriented concepts.

You are required to submit the following files: Java codes in a ZIP file, a Word and PDF report, an MP4 video, and a PowerPoint (PPT) presentation used in the video.

Timeline

Week 10, **Friday**, **May 3, 2024**, **13:00** CST	CW3 is released (This task sheet, skeleton codes, sample test cases)
Week 13, **Sunday**, **May 26, 2024**, **23:59** CST	CW3 Java source code files, video files (MP4, PPT), and report files (Word, PDF) are due
Late Submission Period	5% lateness penalty per-day Max 5 days (Monday-Friday)
Reading Week, **Friday**, **May 31, 2024**, **23:59** CST	End of **Late Submission** Period No submissions are accepted thereafter

Outline

The remainder of the task sheet will outline the game rules, provide detailed specifications of the tasks, and specify the deliverables you are required to submit. Additionally, there is a marking rubric provided to guide you in achieving the best possible outcome.

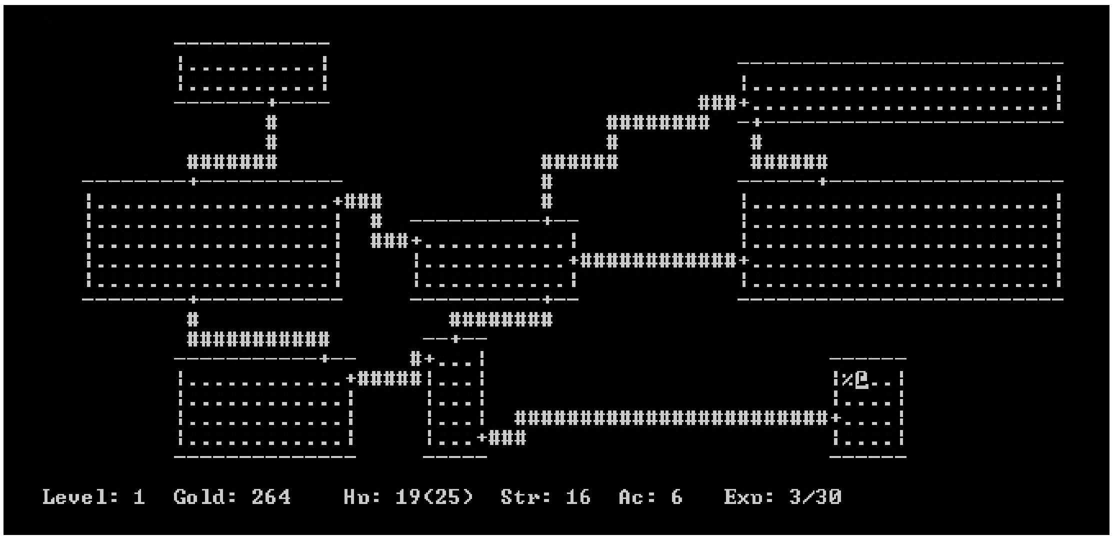
Coursework 3 – Intelligent Rogue Chars

Intelligent game-playing characters have been used in the game industry to harness the power of graph algorithms to navigate complex virtual environments, strategically analyzing interconnected nodes and edges to optimize their movements and decision-making processes. By leveraging graph traversal techniques such as breadth-first search (BFS) and depth-first search (DFS), these characters can efficiently explore vast game worlds, identify optimal paths, and anticipate opponent movements.

In this coursework, you will use graph algorithms you have learned in **Lecture 10** to develop an effective approach to track and intercept a moving opponent in a (simplified version of the) 2D Game called **Rogue**. You will also create a viable plan to evade interception from said opponent.

Rogue

The game Rogue was created by Michael Toy and Glen Wichman, who, while experimenting with Ken Arnold's C library named curses in the late 1970s, designed a graphical adventure game shown below.



```
Level: 1  Gold: 264    Hp: 19(25)  Str: 16   Ac: 6   Exp: 3/30
```

In 1984, CMU graduate students developed Rog-o-matic, an automated Rogue player, which became the highest-rated player. Rog-o-matic's algorithm prioritized avoiding monster encounters to facilitate health regeneration, posing an intriguing graph search challenge, which inspired this coursework.

Rules of the Game

The game of Rogue is played on an N-by-N grid that represents the dungeon. The two players are a rogue and a monster. The **rogue** is represented with the character @. The **monster** is represented by an uppercase letter A through Z.
The monster and rogue take turns making moves, with the monster going *first*. If the monster intercepts the rogue (i.e., occupies the same site), then the monster kills the rogue, and the game ends. In each turn, a player either remains stationary or moves to an adjacent site.

There are three types of sites:

1. Rooms represented by .
2. Corridors represented by +
3. Walls represented by (a space)

The movement rules are:

- If a player is in a room site, then they can move to an adjacent room site in one of the 8 compass directions (N, E, S, W, NW, NE, SW, SE) or a corridor site in one of the 4 directions (N, E, S, W).
- If a player is in a corridor site, then they can move to an adjacent room or corridor site in one of the 4 directions (N, E, S, W).
- The walls are impenetrable.

Consider the following two dungeons.

```
          + + + + +
          +       +
. . . . . . . .   +
. . . . . @ . .   +
. . I . . . . .   +
. . . . . . . .   +
. . . . . . . .   +
          +       +
          +       +
          + + + + +
```

```
. . . . . . . . A .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . @ . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
```

In the first dungeon above, the rogue can avoid the monster I indefinitely by moving N and running around the corridor.

In the second dungeon, the monster A can use the diagonal moves to trap the rogue in a corner.

Monster's Strategy

The monster is tenacious and its sole mission is to chase and intercept the rogue. A natural strategy for the monster is to always take one step toward the rogue. In terms of the underlying graph, this means that the monster should compute a shortest path between itself and the rogue, and take one step along such a path. This strategy is not necessarily optimal, since there may be ties, and taking a step along one shortest path may be better than taking a step along another shortest path.

Consider the following two dungeons.

```
    + + + + + + + +                       + + + + +
    +             +                       +       +
. . . . . . .     +             . . . . . . . .   +
. . . . . . .     +             . . . . . . . .   +
. . . . @ . .     +             . . C . . @ . .   +
. . B . . . .     +             . . . . . . . .   +
. . . . . . .     +             . . . . . . . .   +
. . . . . . . + + +                       +       +
. . . . . . .                             +       +
. . . . . . .                             + + + + +
```

In the first dungeon above, monster B's only optimal strategy is to take a step in the NE direction. Moving N or E would enable the rogue to make a mad dash for the opposite corridor entrance.

In the second dungeon, the monster C can guarantee to intercept the rogue by first moving E.

Your first task is to implement an effective strategy for the monster. To implement the monster's strategy, you may want to consider using BFS.

Rogue's Strategy

The rogue's goal is to avoid the monster for as long as possible. A naive strategy is to move to an adjacent site that is as far as possible from the monster's current location. That strategy is not necessarily optimal.

Consider the following two dungeons.

```
+ + +
+   +
+ + +
  . . . .
  . . . .
  . . . .
  . . . . + + + + +
  @ . . . +       +
  . . . . + + + + +
  . . . F
```

```
. . . . . . .
. . . . . . .
. . . . . @ .
. . . . . . . + + +
. . . . . . .     +
. . . . . . . + + +
. . . . . . J
. . . . . . .
. . . . . . .
. . . . . . .
```

It is easy to see that that strategy may lead to a quick and unnecessary death, as in the second dungeon above where the rogue can avoid the monster J by moving SE.

Another potentially deadly strategy would be to go to the nearest corridor. To avoid the monster F in the first dungeon, the rogue must move towards a northern corridor instead.

A more effective strategy is to identify a sequence of adjacent corridor and room sites which the rogue can run around in circles forever, thereby avoiding the monster indefinitely. This involves identifying and following certain cycles in the underlying graph. Of course, such cycles may not always exist, in which case your goal is to survive for as long as possible. To implement the rogue's strategy, you may want to use both BFS and DFS.

Implementation and Specification

In this section, you will discover the expected details regarding the implementation and specifications of the Rogue game, which you are required to adhere to.

Dungeon File Input Format

The input dungeon consists of an integer N, followed by N rows of 2N characters each. For example:

```
10
    + + + + + + + +
    +             +
. . . . . . .     +
. . . . . . .     +
. . . . @ . .     +
. . B . . . .     +
. . . . . . .     +
. . . . . . . + + +
. . . . . . .
. . . . . . .
```

A room is a contiguous rectangular block of room sites. Rooms may not connect directly with each other. That is, any path from one room to another will use at least one corridor site.
There will be exactly one monster and one rogue, and each will start in some room site.

You will be given 18 dungeon files to test your code with.
You may create your own dungeon files (explain the novelty in your report/video!).
In the rubric subsection below, you are required to show the correctness and the performance of your rogue and monster on **at least 5 non-trivial dungeons** in total.

Game of Rogue Specification

We will provide some files that are already completed as the game infrastructure. There are two files to complete: `Monster.java` and `Rogue.java`, for which some skeleton code is provided.
The given files are only for a quick start: you should modify the files so your program exhibits more object-oriented programming principles!

The following is the interface of `Monster.java` :

```
public Monster(Game g)       // create a new monster playing game g
public Site move()           // return adjacent site to which it moves
```

And the analogous program `Rogue.java`:

```
public Rogue(Game g)        // create a new rogue playing a game g
public Site move()          // return adjacent site to which it moves
```

The `move()` method should implement the move of the monster/rogue as specified by the strategy that you have created.

`Game.java` reads in the dungeon from standard input and does the game playing and refereeing. It has three primary interface functions that will be needed by `Rogue.java` and `Monster.java`.

```
public Site getMonsterSite()      // return site occupied by monster
public Site getRogueSite()        // return site occupied by rogue
public Dungeon getDungeon()       // return the dungeon
```

`Dungeon.java` represents an N-by-N dungeon.

```
public boolean isLegalMove(Site v, Site w)  // is moving from site v
                                               to w legal?
public boolean isCorridor(Site v)      // is site v a corridor site?
public boolean isRoom(Site v)          // is site v a room site?
public int size()                      // return N = dim of dungeon
```

`In.java` is a library from Algorithms optional textbook to read in data from various sources. You will have to create your own input library for your CW3 program.

`Site.java` is a data type that represents a location site in the N-by-N dungeon.

```
public Site(int i, int j)     // create new Site for location (i, j)
public int i()                       // get i coordinate
public int j()                       // get j coordinate
public int manhattan(Site w)         // return Manhattan distance
                                        from invoking site to w
public boolean equals(Site w)        // is invoking site equal to w?
```

If you have two sites located at coordinates (i_1, j_1) and (i_2, j_2), then the Manhattan distance between the two sites is $|i_1 - i_2| + |j_1 - j_2|$. This represents the length you have to travel, assuming you can only move horizontally and vertically.

You should modify the files or create other Java files, **so your final program exhibits more object-oriented programming principles**. Finally, you must only use libraries that are covered in CPT204 (including in Liang textbook). Violating this by using libraries that are not covered in CPT204 will result in **an automatic total mark of 0**.

Deliverables

In this section, you will find the details regarding the files that you are required to submit, and the report and video specifications.

Submission Requirements

You are required to submit:

1) The Rogue.java and Monster.java source code Java files, as well as any other auxiliary Java files and the dungeon files that you selected/created in your project, altogether combined in a ZIP file.
2) Your report, in both Word and PDF format (2 separate files).
3) The PPT of your video presentation.
4) The video recording of your presentation in a MP4 file.

The submission deadline is **Sunday, May 26, 2024, at 23:59 CST**.

You may submit late, with a maximum grace period of 5 days, during which a 5% lateness penalty will be applied for each late day. Therefore, after **Friday, May 31, 2024, at 23:59 CST**, no submissions will be accepted.

Report Requirements

Write a report satisfying the following criteria:

1. The purpose of your report is to explain your code, algorithms, algorithm analysis, and OOP elements in **well-detailed** manner.
2. Your report must consist of **exactly 6 Chapters**:

   Chapter 1 – Object-oriented Principles

   (you may add subchapters here)

   Chapter 2 – Monster Algorithm

   Chapter 3 – Rogue Algorithm

   Chapter 4 – Monster Algorithm Analysis

   Chapter 5 – Rogue Algorithm Analysis

   Chapter 6 – My Java Code

   For more details on the contents of each section, you should refer to the Rubric on page 11.

3. You must include **all** your code in Chapter 6 as a text, copy paste each source file content into the report.
   You must **not** use screenshots in Chapter 6.
   Using screenshot in Chapter 6 will result in **an automatic total marks of 0**.
   (you may use screenshot in other chapters)
4. Write your report using Word with the following setting:
   Font **Calibri**, Font Size **12**, Line Spacing **1.5**, Normal Margins.
   The page limit is a maximum of **20 pages**, *not* including Chapter 6.
5. Consider using images and diagrams to improve the readability of your report. Please refer to the rubric in the following subsection.
6. Save your Word document as a PDF.
   Submit to Learning Mall Assignment Box **both** the Word document file and the PDF file.

Video Requirements

Create a PPT presentation and video explanation using the PPT satisfying the following requirements:

1. The purpose of your video presentation is to explain your code, algorithms, and OOP design in a **succinct** manner.
2. You can use any template for your PPT, **not** limited to the XJLTU standard theme.
3. The length of the video must be **less than or equal to 8 minutes**.
   Violating the video length requirements will result in **a total marks of 0** for your coursework mark.
4. Your video **must display your face** and **include your audio** for the purpose of authenticity verification.
   Do **not** use English audio translation software to narrate your video.
   Violating the requirement to show your face and use your voice will result in **a total marks of 0** for your coursework.
5. The clarity of the presentation will be graded. Please refer to the rubric in the next subsection.

6. Submit to Learning Mall Assignment Box **both**:
    a. The video file in **MP4** format,
    b. The **PPT** file you used to create a video.

Equal Contribution Requirements

In adherence to academic integrity and fairness, it is imperative that both team members contribute equally to the completion of the coursework:

1. Each member should actively participate in the development process, ensuring a balanced distribution of workload and responsibilities.
   For example, one team member may mostly undertake the coding of the rogue character while the other focuses on the monster character, and they may divide tasks such as testing, documentation, report writing, and presentation preparation equitably.
2. **Both** team members must show their faces in the video, one at a time.
3. Should there be any concerns regarding unequal contributions, team members are encouraged to notify the module leader promptly to facilitate appropriate assessment and marking decisions, to ensure transparency and accountability.

Rubric

Criteria	Description	Marks
Object-Oriented Principles	Evaluation of the effective use of object-oriented principles (encapsulation, inheritance, polymorphism, abstraction) in the overall Java program solution.	25
Code Clarity and Readability	Assessment of code clarity, organization, and readability. Includes appropriate variable naming, comments, documentation, and code structure.	5
Monster and Rogue Demo	Evaluation of the correctness and optimality/sub-optimality of your Monster and Rogue algorithms. Write in report, and show demo and argue in video on ***at least 5 non-trivial dungeon files*** in total for both Monster and Rogue characters. Justify your strategies, explaining their strengths and also weaknesses.	20
Monster Algorithm Analysis	Examination of how graphs algorithms are implemented in your Monster Algorithm, and their efficiency analysis.	15
Rogue Algorithm Analysis	Examination of how graphs algorithms are implemented in your Rogue Algorithm, and their efficiency analysis.	15
Report Clarity, Structure and Presentation	Assessment of the report's clarity, structure, organization, visual elements, and overall quality of writing. Includes language use and presentation of data structure usages.	10
Video & PPT Clarity, Delivery and Engagement	Assessment of the video presentation's delivery, clarity and engagement. Includes speaking clarity, engagement with the audience, and visual elements in demonstrating the execution of the software.	10
Total Marks		100

Show that you satisfy **all** the rubric components in **both** your report and video!

Please note again that using libraries **not** covered in CPT204; using screenshots or not including all your codes in *Chapter 6* of the report; or submitting a video of length **longer** than *8 minutes* or **without** your face/voice will result in an automatic **total marks of 0**.

Acknowledgement

This coursework is adopted from an assignment from the Algorithms (an optional Textbook of this course) with thanks to Robert Sedgewick, Kevin Wayne, and Andrew Appel.

Academic Integrity

1. Plagiarism, e.g. copying materials from other sources without proper acknowledgement, copying, or collusion are serious academic offences. Plagiarism, copying, collusion, using or consulting unauthorized materials (including code sharing forum, and generative AI tools including, but not limited to, ChatGPT) will not be tolerated and will be dealt with in accordance with the University Code of Practice on Academic Integrity.
2. In some cases, individual students may be invited to explain parts of their code in person, and if they fail to demonstrate an understanding of the code, no credit will be given for that part.
3. In more severe cases, the suspected violation will be directly reported to the Exam Officer for further investigation and, if confirmed, will be permanently recorded in the offender student's official academic transcript.

Please read: https://academicpolicy.xjtlu.edu.cn/article.php?id=98

Question Forum

If you have questions regarding Coursework 3, please post your questions in CW3 Forum: https://core.xjtlu.edu.cn/mod/forum/view.php?id=137339

This is the end of CPT204-2324 CW3 Task Sheet.