CPT204 2223 Final Project Task Sheet 1

Overview

Final Project, the final coursework assessment component this semester, consists of 2 parts: Part A and Part B. It contributes 40% to your final marks.

In **Part A (200 marks)**, you will work on creating a game using the data structures you learned in the course. After that, more interestingly, you will work in a team of three to create an AI player to play that game. In **Part B (200 marks)**, you will have to individually solve two problems of various data structures, problem-solving and object-oriented techniques that are derived from Lecture 1-14 and Lab 1-13.

Timeline

Week 11, Tuesday, **April 25, 2023**	Final Project is released (Task Sheet 1 & 2, Skeleton Codes, Partial Test Cases, Demo Video)
Week 14, Tuesday, **May 16, 2023, 11:00 CST**	Part A.2 Dropboxes for mp4, ppt, and code are ***open***
Week 14, Friday, **May 19, 2023, 11:30 CST**	**Submission Day**: Part A.1 and Part B.1, B.2 submissions are ***open***
Week 14, Friday, **May 19, 2023, 12:30 CST**	Part A.1, A.2 Java source file, video files (mp4, ppt) and Part B.1, B.2 submission are ***close***
Late Submission Period	5% lateness penalty per-day for video files (mp4, ppt) Maximum 5 working days
Exam Week, Friday, **May 26, 2023, 12:30 CST**	End of Late Submission Period for Part A.2 video files No video files submissions are accepted thereafter

Lateness Policy

Video, ppt, code of Part A.2 are allowed to have late submission with penalty.

There will be **no** late code submission for Part A.1 and Part B.1, B.2 since feedback is given, to be consistent with University lateness policy on assessment with feedback.

Outline

The rest of the task sheet will describe the two parts of the final project, and the Submission Day in detail.

Final Project Part A – Ataxx on AI

Part A Overview

In Part A, you are going to design a game called Ataxx, and an AI to play the game against a human player automatically. Firstly, you will *work on your own* to complete the basic Ataxx game – played by two human players manually taking turn. Next, you will work in *a team of three students* to research and design your own AI player which should be good enough to defeat average players! If you still have enough time, you can complement your work by designing a good GUI for some bonus points.

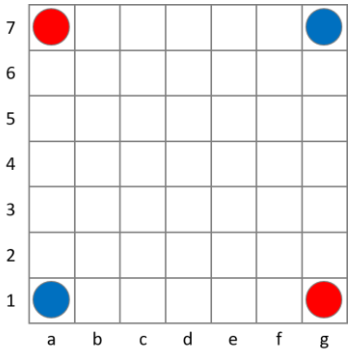
Ataxx was invented in 1988 by David Crummack and Craig Galley (originally named Infection) as a board game with very simple rules to be played against a computer [1]. Several methods [1-4] have been developed to leverage AI algorithms and heuristics in making the computer a better player. It appeared and gained popularity in the 90s in PC and Arcade Games with names such as Microscope, Show of Infection, SlimeWar or Frog Cloning [5].

Basic Ataxx

Ataxx is a 2-player board game played on a 7-by-7 board. Let us call the players **Red** and **Blue**. They place their corresponding pieces on the squares of the board in turns. The rows are denoted by **1**-**7** and columns **a**-**g**. Each square is filled by a Red or Blue piece or empty. Initially, player Red will make the first move.

Here is the configuration of **the initial board**:

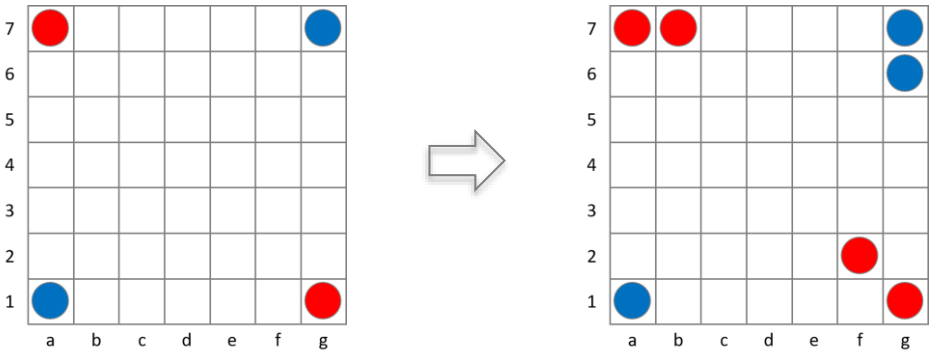
- Red in top left and bottom right (**a7** and **g1**).
- Blue in bottom left and top right (**a1** and **g7**).



The initial board

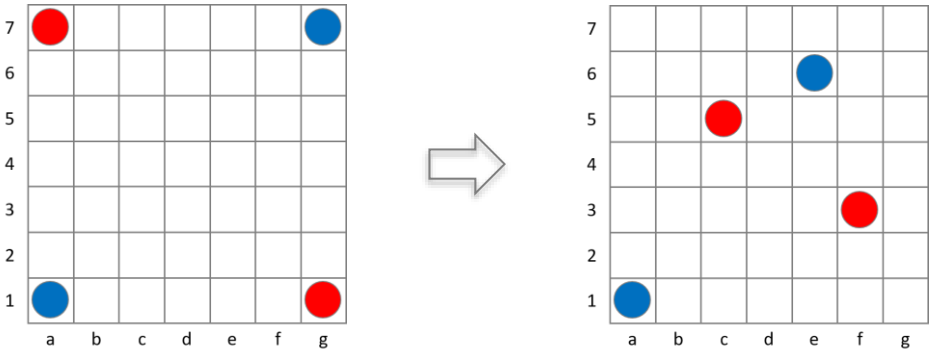
There are two kinds of moves in Ataxx:

1. **Clone**: place a new piece of your color next to an existing piece of your color – horizontally, vertically, or diagonally adjacent.



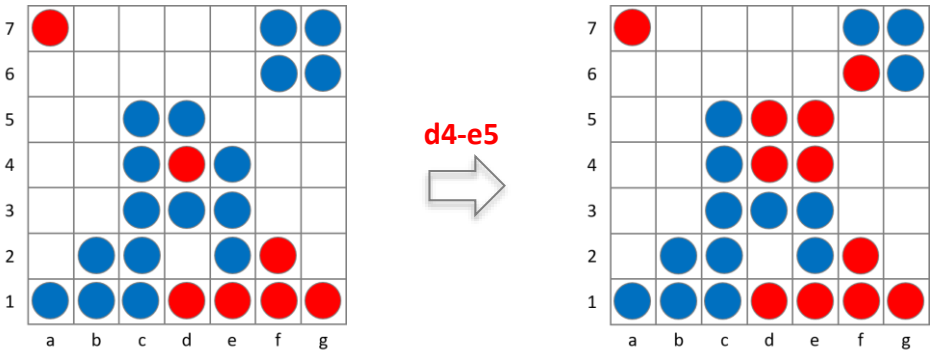
For example, three clone moves from the initial board **a7-b7**, **g7-g6**, and **g1-f2** will result in the board on the right.

2. **Jump**: move a piece of your color to a non-adjacent square that is at most two rows and two columns away.

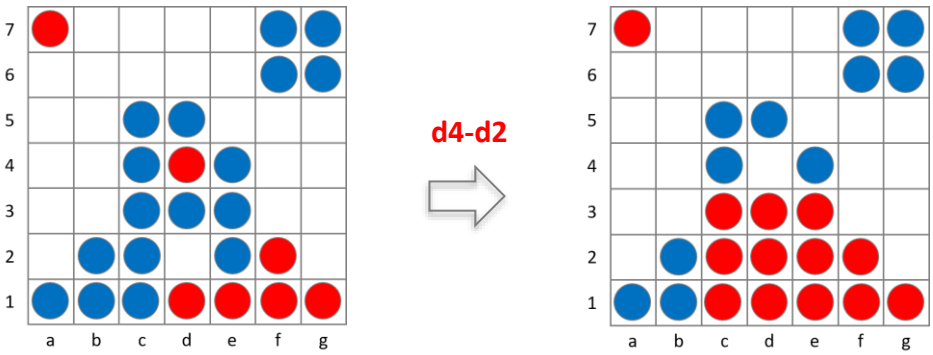


For example, the board on the right is the result of three jump moves **a7-c5**, **g7-e6**, and **g1-f3** from the initial board.

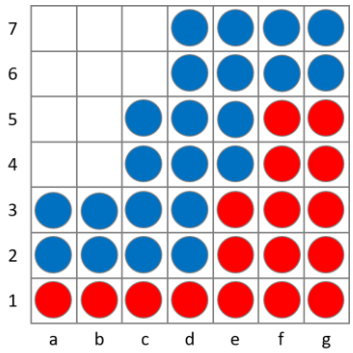
When you are making a move, ***all*** pieces with opposite color next to the moved piece are replaced by pieces of your color.
For example, the next Red clone will produce the board on the right as follows:



And another example, the next Red jump will have the following result:



Sometimes a player cannot move and must **pass**, like the Red player below:



A player can only pass if they cannot move (they must clone or jump if possible).

The following are the conditions that **end** the game:

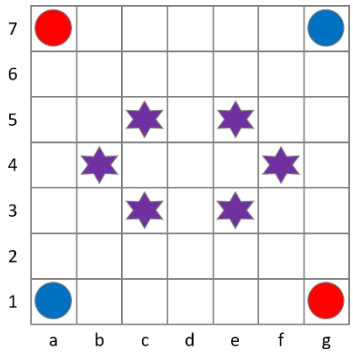
- Neither player can make a move (i.e. the board is full).
- A player has no pieces of their color on the board.
- There are 25 *consecutive* jumps without any clones.

When the game ends, the player with the **most** pieces on the board wins. If there are the same number of red and blue pieces on the board, then the game results in a **tie**.

Finally, we can also have **blocks** with the following rules (error message if violates):

1. Blocks must be placed in the beginning before either player makes a move.
2. Additional blocks that are reflections of that block around the middle row and middle column are *automatically* added.
3. Pieces *cannot* be placed on these blocks, other blocks cannot either.

For example, placing blocks at **c5** and **b4** at will result in (others automatically added):



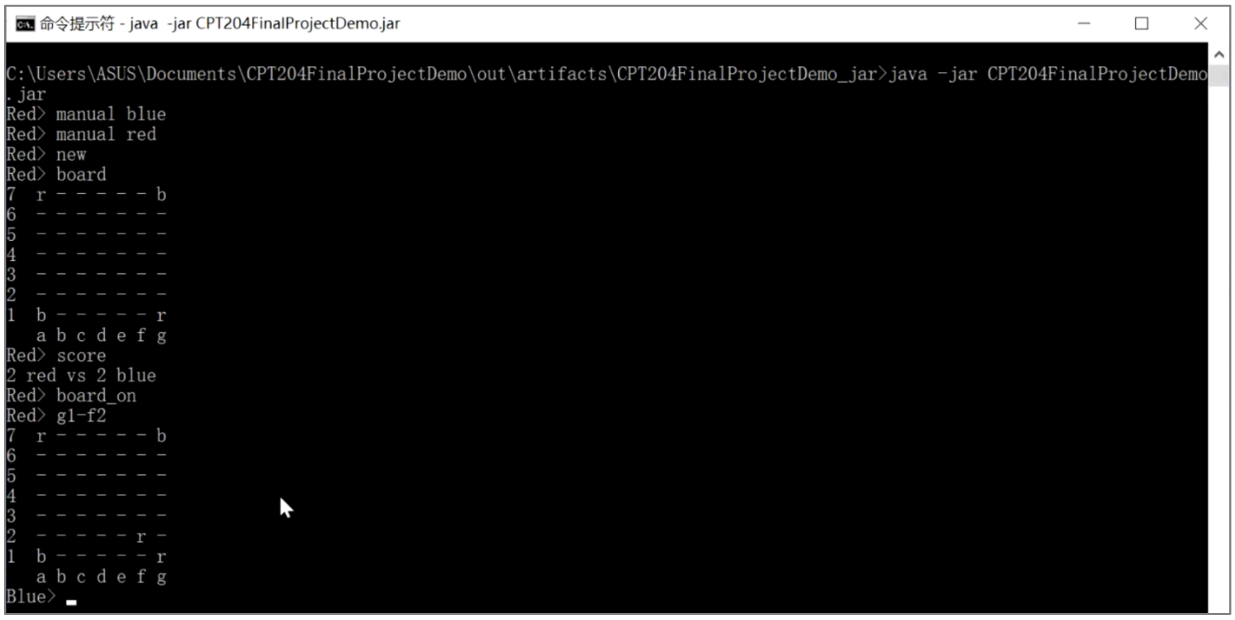
Part A.1 Basic Ataxx

Your first task is to implement the **logic** of Ataxx, to enable operations such as:

- Create a new game and set the initial board and blocks.
- Display the board in a text-based interface.
- Implement clone and jump moves, checking first whether they are allowed.
- Make a move and update the board accordingly.
- Determine whether pass is needed, or the game has ended – then report the result (i.e. identify the winner or declare a tie).

You will find **the details** in **Task Sheet 2 pdf**.

You are expected to complete a skeleton code that runs the Ataxx game in a **text-based interface**, such as in the following Windows Command Prompt:



The prompt indicates the current player (Red or Blue), and the program uses the following format to print the board:

```
7  r - - - - - b
6  - - - - - - -
5  - - X - X - -
4  - - - - - - -
3  - - X - X - -
2  - - - - - r -
1  b - - - - - r
   a b c d e f g
```

where: `-` indicates an empty square, `r` indicates a red piece, `b` indicates a blue piece, and `X` indicates a block.

The commands are typed into a prompt manually by the two manual players, and for Part A.1, you are expected to implement these commands:

- **`new`** : ignores the current game if one is in progress, clears the board to the board initial configuration, and starts a new game.
- **`quit`** : abandons the current game and exits the program.
- **`manual <red/blue>`** : sets the player with that color to be a manual player.
- **ai <red/blue>** : sets the player of that color to be an AI player.
- **score** : prints out the current number of red and blue pieces on the board, such as `"2 red vs 2 blue"`.
- **`block <xy>`** : sets a block in the square at location column `x` and row `y` (e.g. `block c3`) and all reflections of that square around the middle row and middle column. Blocks cannot be placed at the corners where the beginning piece are. Blocks may not be placed after a piece move. They can only be placed in the beginning of the game, or after the board is cleared. In case of any violations, the command has no effect.
- **`c0r0-c1r1`** : the current player makes a move from location `c0`, `r0` to location `c1`, `r1`. The program should automatically detect if the move was a clone or a jump, and update the board accordingly.
- **`board`** : prints out a board in the format above.
- **`board_on`** : causes the board to get printed after each move.
- **`board_off`** : causes the board to no longer get printed after each move.

Video demo is given to you in the zip file, please watch to see how the commands are executed and what their output are.

If any of the game end conditions is satisfied, according to the rules explained above, the program should print one of the following:

```
* Red wins!
* Blue wins!
* Draw!
```

Part A.1 Details and Deliverables

Find more details about Part A.1 in CPT204_2223_Final_Project_Task_Sheet_2.pdf.

During the Submission Day, you are supposed to complete a coding quiz that contains multiple methods related to the empty parts of the skeleton code.
You can find the details in the Submission Day section of this task sheet.

The total marks for all the methods in part A.1 is **100 points**, for passing all tasks and all the test cases. Partial marks will be given for passing some tasks and some test cases.

Part A.2 Ataxx on AI

Your second task is to research and implement an **AI player** in AIPlayer Class (in AIPlayer.java) that you can play the game against, with the following specifications:

- The AI needs to be able to force a win if possible within four moves of a given position (e.g. it cannot just be fully random), for full correctness marks.
- The AI must use no more than 5 seconds to make a move (e.g. it cannot exhaustively check the huge space of all moves).
- You may introduce new program output such as printing the AI player moves.
- You may use any algorithms or heuristics to implement your AI, especially the algorithms that you have learned in your Year 2 and Year 3 modules, but not restricted to them – you may invent your own!
  Make sure you have first researched the related literature as there has been published methods on this topic. Bright uses Negamax and Alpha-Beta Pruning [2]; and Cuppen uses Monte-Carlo techniques [3] in creating an Ataxx AI.
  You need to cite whenever needed.

Part A.2 Deliverables

Create a ppt, a video, and make a submission to Learning Mall Dropbox before/during the Submission Day with the following requirements:

1. The ppt and video must contain **discussions of Object-Oriented Programming Principles** and **the AI algorithm** that you use to implement the AI player.
   You can find the details in the next subsection.
2. The length of the video must be **less than or equal to 6 minutes**.
   Violating the length requirements will result in **0 marks** of your A.2 grade.
3. Your video **must show at least one of the team member's face** for the purpose of authenticity verification.
   Violating this requirement will result in **0 marks** in your A.2 grade.
4. You may want to make your video look nicer, however, the grade will not be based on the looks. Only the quality and clarity of the discussion will count.
   A simple recording of a ppt explanation while showing the presenter face in a box by shared screen with BBB, Zhumu, or Tencent Meeting would be sufficient.
5. Submit to Assignment Dropbox in Learning Mall the following:
   a. The video file in **mp4** format.
   b. The **ppt** file you used to create the video.
   c. A **zip** file that contains **all your code** for further verification.
6. Three dropboxes will open from **Tuesday, May 16, 2023, 11:00 CST** and close on **Friday, May 19, 2023, 12:30 CST**.

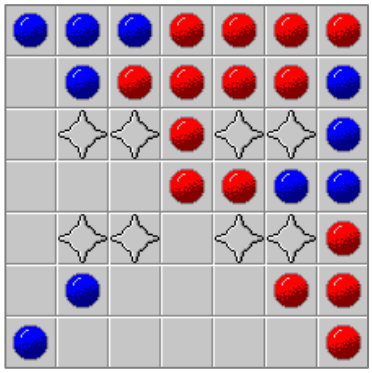
Marking Rubric

Criteria	Marks	Total
Applying Object-Oriented Principles		50
Encapsulation, Inheritance and/or Polymorphism	10	
API and New Commands Implementation	10	
Modularity, Data Hiding via Multiple classes	10	
Assertion and Error Handling	10	
Coding style and Class, Method Specification via Javadoc	10	
AI Algorithm		30
Correctness of Efficient Algorithm Implementation	10	
Experiment Result and its Analysis	10	
Novelty	10	
Clarity		20
Slides Clarity	10	
Presentation Clarity	10	
		100

The total marks for part A.2 is **100 points**, for both the video presentation and the ppt.

Optional Part: Ataxx GUI

As an optional task, you may also complete the **GUI Class** (in GUI.java) to enable the game to be played in a more interesting interface.
For example, GUI from [1]:



You need to explain your GUI design decisions and code in the video and ppt to be submitted along with Part A.2 Deliverables.

Optional Part Deliverables

You need to **add** more materials to the Part A.2 Deliverables:

1. The ppt and video must also contain **demo and discussions of the GUI design** that you use to implement your Ataxx game interface.
2. After adding, the length of the video must be **less than or equal to 7 minutes (one minute added)**.
3. Submit to Assignment Dropbox (the same as Part A.2) in Learning Mall the following:
   a. The video file in **mp4** format.
   b. The **ppt** file you used to create the video.
   c. A **zip** file that contains **all your code** for further verification.
4. Three dropboxes (the same as Part A.2) will open from **Tuesday, May 16, 2023, 11:00 CST** and close on **Friday, May 19, 2023, 12:30 CST**

The total marks for this optional part is **25 points**.

Marking Rubric

The maximum marks of your submission:

1.	Part A.1 Basic Ataxx Methods Correctness	**100 marks**
2.	Part A.2 AI Algorithm, OOP Design, and Video Clarity:	**100 marks**
	Total	**200 marks**
3.	Optional GUI Bonus	**25 marks**
	Max Total	**225 marks**

References

[1] http://www.pressibus.org/ataxx/indexgb.html
[2] Bright, C., 2010. CS 686 Programming Project: Ataxx with AI.
[3] Cuppen, E., 2007. Using monte carlo techniques in the game ataxx (Doctoral dissertation, B. Sc. Thesis, Maastricht University).
[4] Michael Levin, SlimeWar: An Ataxx Clone, http://sourceforge.net/projects/slimewar/
[5] Ribeiro, P., Simões, H. and Ferreira, M., 2009. Teaching artificial intelligence and logic programming in a competitive environment. Informatics in Education, 8(1), pp.85-100.

Final Project Part B – Short Coding Question

In part B, you are going to solve **2 coding questions**. They are closely related to the works you have done throughout the semester in Lab 1-13. The questions are written in autograded Learning Mall Quiz. You will be allowed to access the course materials similar to an open-book exam setting. You will be given two random questions from a question bank with similar difficulties. Your code will be graded on **new testcases** later. More details will be given via **Learning Mall Announcements**.

While the specific questions are not going to be revealed here, the range of topics will be given below. You can also practice by reviewing all your works in Lab 1-13. You will be able to access the questions in the Learning Mall Quiz on the Submission Day: Friday, 19 May 2023, 11:30 – 12:30 CST. There will be **no** late code submissions.

The marks for each question in part B is **100 points**, for a total of **200 points**.
Partial marks will be given for passing some test cases, after graded on a new set.

Data Structures

List, ArrayList, MyList, SLList, DLList, ARList
Deque, LLDeque, ARDeque
Map, HashMap, HAMap
Set, ARSet, HASet
MinPQ, ARBinHeap
Union Find, Quick Find, Quick Union, Weighted Quick Union
Generic Data Structure of the above and their subclasses

Object-oriented Features and Problem-solving Techniques

Empty Constructor, Default Constructor, Copy Constructor, Deep Copy
Iterative, Recursive, Recursion with Helper Method
Mutates, Not Mutate, Immutable
Resizing Array, Table Doubling/Halving
Checked/Unchecked Exception, Assertion
Iterator, Iterable, Enhanced For Loop, ToString
Interface, ADT, Interface Inheritance, Implementation Inheritance, Casting
Static/Dynamic Type, Dynamic Method Selection, Overloading, Overriding
Equality, Higher Order Functions, Comparator, Comparable, HashCode

Final Project – Submission Day

The submission day is on **Friday, May 19, 2023**.

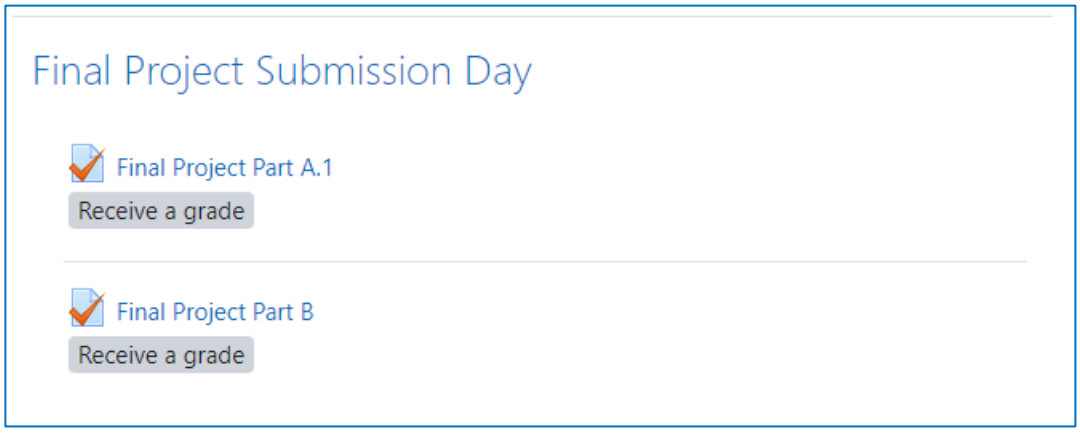
Prepare your laptop, with all necessary software installed. Make sure you have a good Internet connection. Have Learning Mall opened in your browser, preferably Chrome or Firefox. Do ***not*** use Internet Explorer.

Connect into a Tencent Meeting Room with your official student account. Meeting ID and other requirements are to be announced later. The password to access the quizzes will be shared in that room.

Submission Day Section on Learning Mall

Part A.1 and B quizzes will be accessible at 11:30 CST on Friday, May 19, 2023. They will be closed at 12.30 CST.

You will see a similar section such as shown below.



Additionally for Part B, you will also see a problem description and class/method specification outlining each problem, similar to our lab sheets. Some may include a skeleton code (in a Java or zip file) as well. You may want to copy and paste the skeleton code into your coding environment, so please have it ready.

Learning Mall Quiz Submission

When Quiz Part A.1 and B open at 11:30 CST, you will be given additional test cases. You may want to check your code against those test cases in your IDE first, *not* in the Learning Mall Quiz.

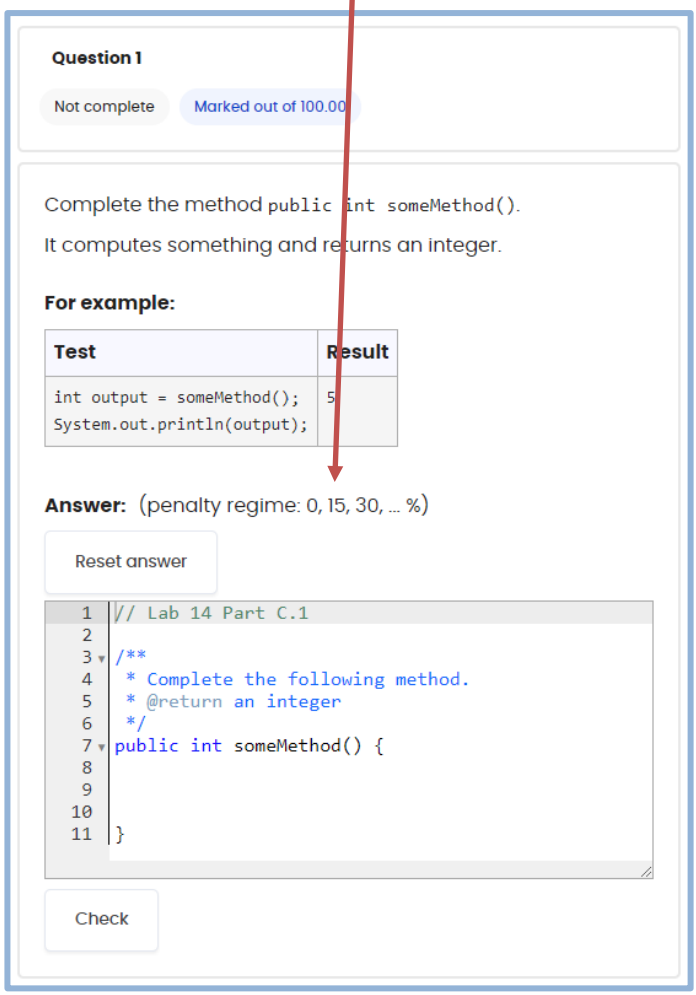
You have 60 minutes to test, debug — if needed, and submit your code. Submit your code before the closing time at 12:30 CST. It is *highly recommended* to submit a few minutes earlier to account for network transmission delay.

For each question, you have to complete the method or the class, as specified in the problem description and specification. You may use *Check* against the test cases multiple times.

In addition, you may include your private helper methods in the submission box.

Adding another elements, such as importing libraries, will result in **0 marks**.

The first *Check* will have no penalty if failing the test cases. The subsequent *Check*s, however, will each incur an additional 15% penalty, cumulatively for each incorrect *Check*. Your code will be graded on a *new* set of testcases and *partial marks* are given.



Useful Tips for Submitting Your Work

1. If you copy and paste code from your IDE, please double check the parenthesis, class/method headers, etc., before clicking the *Check* or *Finish Attempt* button.
2. For Part A.1 and Part B Quiz, ***do not go to the next page*** before completing your submission. You ***cannot*** **go back after clicking Next**.
3. Test your code intensively before submitting your work.
4. Familiarize yourself with the error messages of the autograder systems that we have seen throughout the semester.

Academic Integrity

1. Plagiarism, e.g. copying materials from other sources without proper acknowledgement, copying, or collusion are serious academic offences. Plagiarism, copying, or collusion 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 violation will be reported to the Exam Officer for further investigation and will be permanently recorded in the student's official academic transcript.

---------- **This is the end of CPT204 2223 Final Project Task Sheet 1** ----------