Outline

- Deadline
- No Plagiarism
- Leave Comments
- Scoring
- Problem 1
- Problem 2
- Submission
- Questions
- Appendix

Deadline

- December 19th (Monday) **23:55** (LearnUs server time)

No Plagiarism

- No Mercy.

- The punishment will be made to both
  - the person who copied the code, and the person who shared the code.

- We will do plagiarism test with codes that were made in previous semesters and also in google. So be careful ☺

Leave Comments

- Leave comments in your file for TAs to understand your code.
- If no comments in the file, there may be a reduction of points.

Scoring

- You should take care of your code not terminating by an issue in the middle of the loop
  - There can be some additional points except for the outputted file results (not always), so please try your best to submit your codes in proper format.

- Each assignment will be scored with total 100 points (there will be some additional points).
  - We will announce you scoring policy after we finish scoring of each assignment

- For every problems, you don't have to worry about the wrong/fault input.

- **Problem1 (20%)**   **Problem2 (80%)**

Final Assignment

- You are going to create a simple game for the final assignment.
- There are a player, enemies, and a goal in the map.
- You can move player to “up”, “down”, “left”, “right” by pressing ‘w’, ‘s’, ‘a’, ‘d’ respectively.
- Enemies will move around the map to patrol their areas.
- Game over if player meets the enemy or get in to the guard area of the enemy.
- Victory if player avoids all enemies and reaches the goal.

Problem 1

- Implement Grid2D class using template.
- Grid2D manages 2D array with arbitrary type.
- See provided “Grid.h” header file, and implement all functions and make complete version of the class.
- You need implement:
  - Grid2D(int width, int height)
  - Grid2D(int width, int height, T defaultValue)
  - T &get(int x, int y)
  - void set(int x, int y, T value)
  - void fill(T value)
- Feel free to add *any additional member variables or functions*.

Problem 1

- Implemented Grid2D will be used for Problem 2.
- Do NOT create additional .cpp file for Grid2D class.
- Instead, you can implement all your codes in “Grid.h” file.
- Note that separating declarations / implementations of the template classes can cause linking error of the compiler. (https://isocpp.org/wiki/faq/templates#templates-defn-vs-decl)

Problem 2

- Implement codes for the main game.
- We provide skeleton codes, including the codes for rendering and the overall game management.
- You need to implement several functions of 7 classes:
  - Game
  - Object
  - Player
  - Enemy, VerticalEnemy, HorizontalEnemy, StandingEnemy

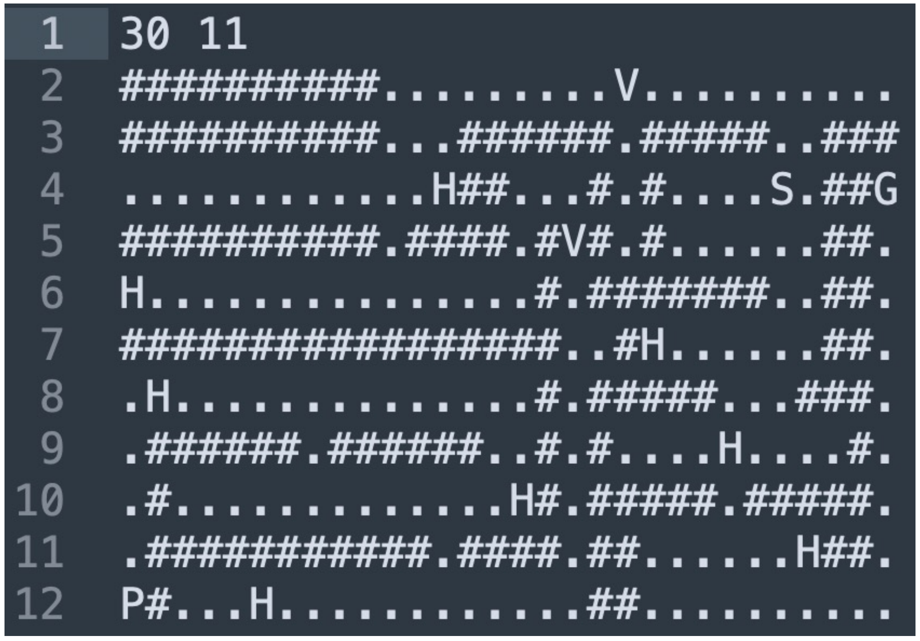
Problem 2: Game class

- In Game class, you need to implement two functions
  - loadMap()
  - runOneStep()

Problem 2: Game class

- loadMap(string filePath)
  - read text file and initialize map, player, enemies and goal.
  - Player and enemies cannot move to the walls.
  - The map should be surrounded by the walls.
  - First line is “width height”, and following lines are the map.
  - If the input map has width = 30, height = 11, the actual map will be width = 32, height = 13. (with surrounding walls)

‘#’ = wall,
‘.’ = empty,
‘P’ = player,
‘H’ = horizontal enemy,
‘V’ = vertical enemy,
‘S’ = standing enemy
‘G’ = goal



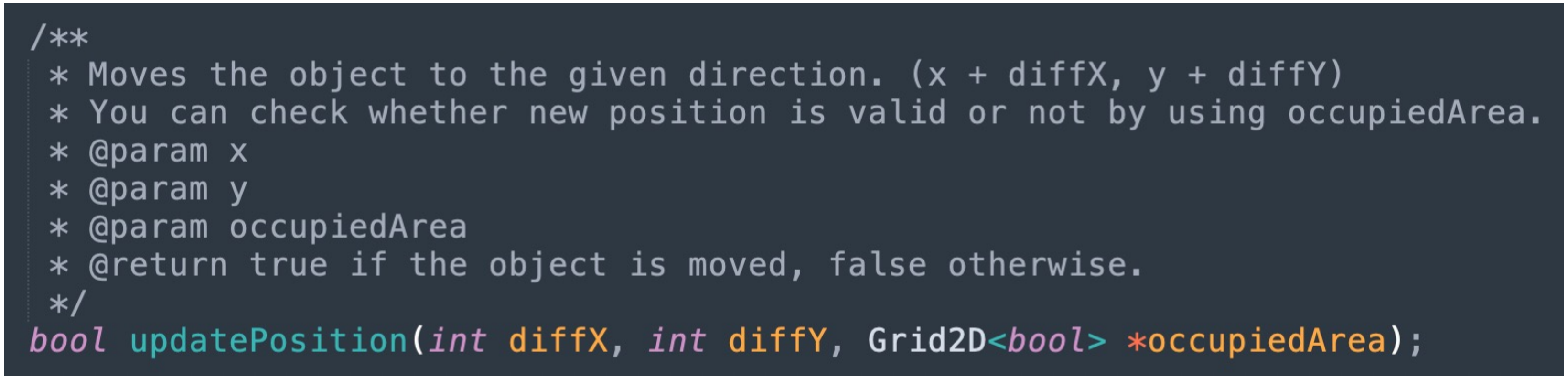
```
30 11
##########.........V..........
##########...######.#####..###
............H##...#.#....S.##G
##########.####.#V#.#......##.
H...............#.#######..##.
#################..#H......##.
.H..............#.#####...###.
.######.######..#.#....H....#.
.#.............H#.#####.#####.
.###########.####.##......H##.
P#...H............##..........
```

Problem 2: Game class

- runOneStep(int command)
    - Handling one step of the game.
    - You should move player based on ‘command’, update the enemies, and check game over or victory.
    - You don’t need to care about rendering. Provided skeleton codes will do that.
    - What you need to do is properly update member variables of Game class.

Problem 2: Object

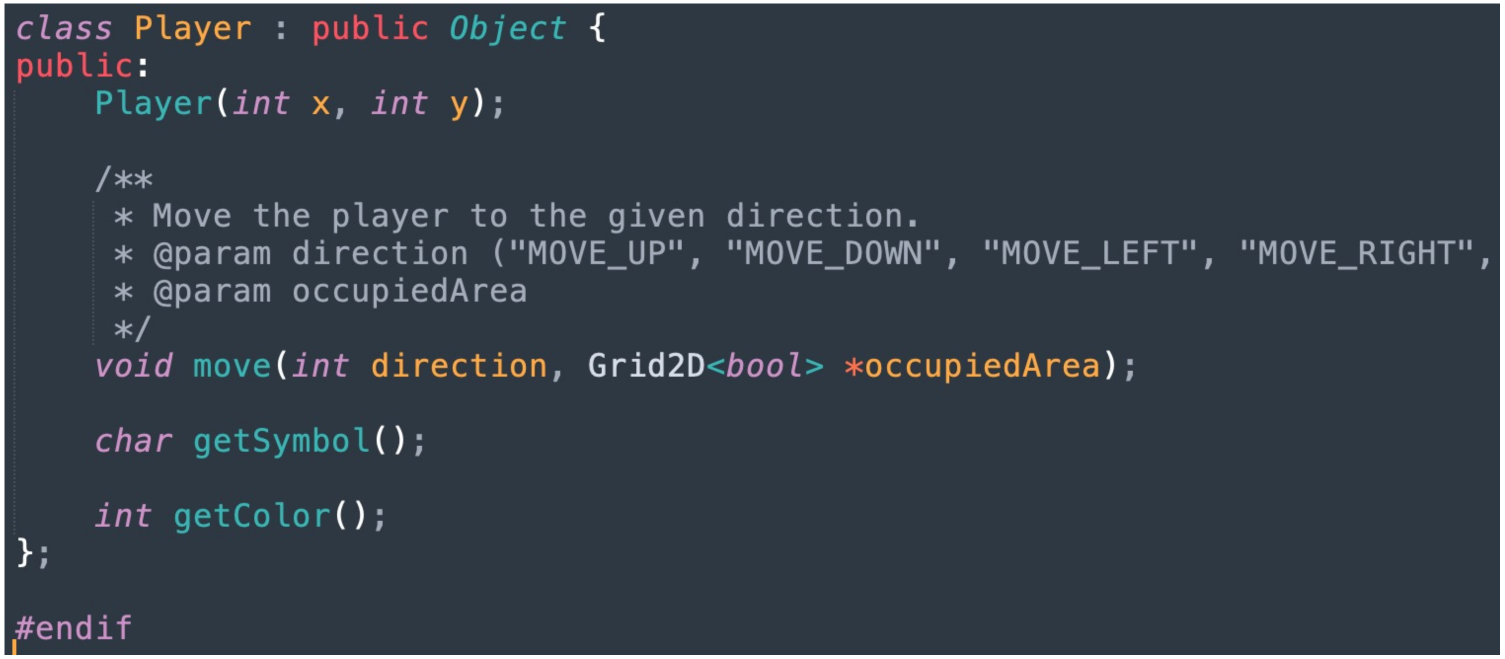
- Object class is an abstract class.
- You need to implement one function, and other virtual functions should be implemented in child classes.



```
/**
 * Moves the object to the given direction. (x + diffX, y + diffY)
 * You can check whether new position is valid or not by using occupiedArea.
 * @param x
 * @param y
 * @param occupiedArea
 * @return true if the object is moved, false otherwise.
 */
bool updatePosition(int diffX, int diffY, Grid2D<bool> *occupiedArea);
```

Problem 2: Player

- Player class is a derived class of Object.
- You need to implement two virtual functions derived from Object, and move() function.
- Player can move up, down, left, right or do nothing.



```cpp
class Player : public Object {
public:
    Player(int x, int y);

    /**
     * Move the player to the given direction.
     * @param direction ("MOVE_UP", "MOVE_DOWN", "MOVE_LEFT", "MOVE_RIGHT",
     * @param occupiedArea
     */
    void move(int direction, Grid2D<bool> *occupiedArea);

    char getSymbol();

    int getColor();
};

#endif
```

Problem 2: Enemy

- Enemy class is derived from Object, which is also an abstract class.

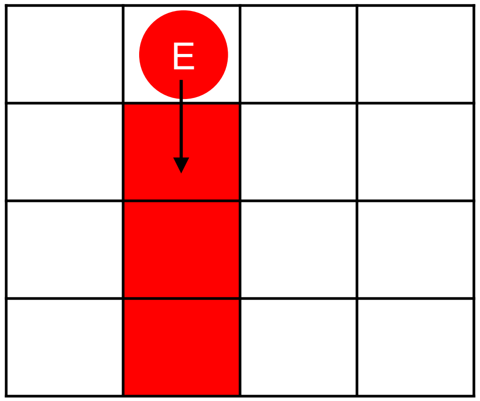
- Enemy has two virtual functions:
  - move(): defines how enemy moves.
  - guard(): defines which areas enemy will guard

- These virtual functions should be implemented in derived classes.

- You need to implement 3 types of enemies: VerticalEnemy, HorizontalEnemy, StandingEnemy

Problem 2: VerticalEnemy

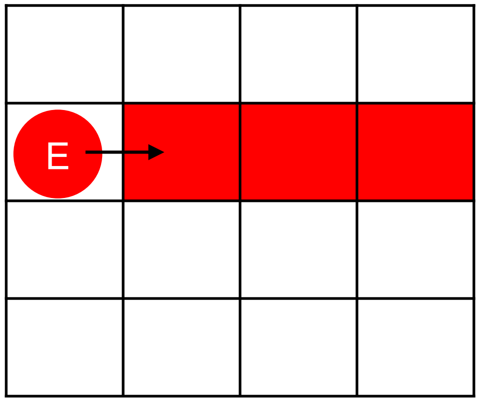
- move()
  - VerticalEnemy moves up or down until it reaches the wall.
  - After reaching the wall, it turns to the opposite direction and moves forward.
  - Its first direction should be “down”.
  - It should move every 3 step.
    (first move should be 3rd step of the game)
- guard()
  - VerticalEnemy guards 3 cells in front of its moving direction.
  - Ex) when it is moving “down”:



Problem 2: HorizontalEnemy

- move()
  - HorizontalEnemy moves right or left.
  - Same as VerticalEnemy, it should move forward until it reaches the wall, and then turn to the opposite direction.
  - Its first direction should be “right”.
  - It should move every step.

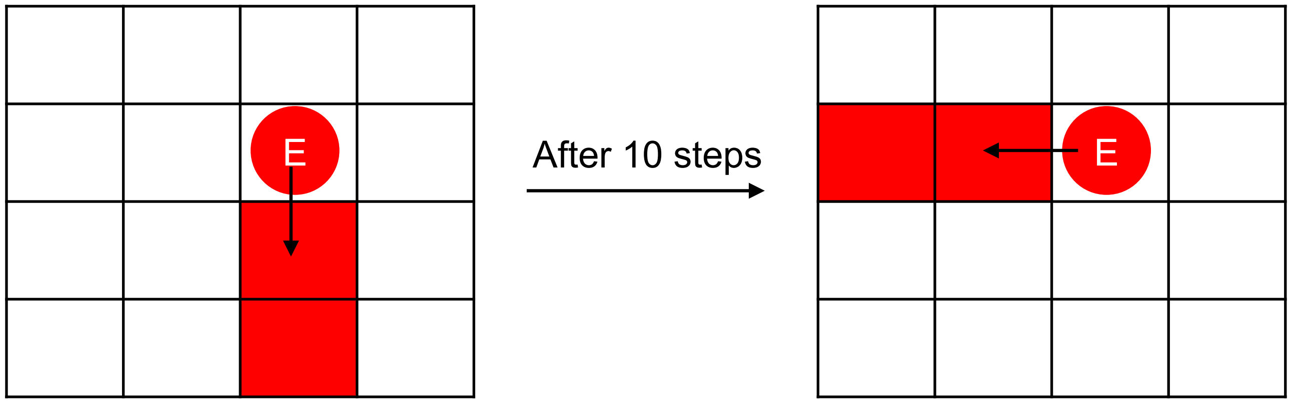
- guard()
  - HorizontalEnemy guards 3 cells, same as VerticalEnemy.
  - Ex) when it is moving “right”:



Problem 2: StandingEnemy

- move()
  - StandingEnemy doesn't move.
  - Instead, it turns 90 degrees at every 10 steps. ("right", "down", "left", "up")
  - Its first direction is "right".

- guard()
  - StandingEnemy guards 2 cells in front of its direction.
  - Ex) when it is in "down" direction:



Problem 2

- You can compile your codes using provided Makefile.
- You need to first install ncurses package with following commands:
  - sudo apt-get update
  - sudo apt-get install -y libncurses5-dev libncursesw5-dev
- Or you can simply install by running “make setup”.
- To compile your main code, run “make”.
- Then you can run your output by “./output”.
- If you have any problem with the message “sudo command not found”, please erase all sudo keywords in your commands including Makefile.

Appendix

- C++ vector
  - https://cplusplus.com/reference/vector/vector/
- Makefile…
  - https://makefiletutorial.com