Assignment Number 3

CL1004 Object Oriented Programming

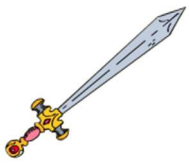
Spring 2025

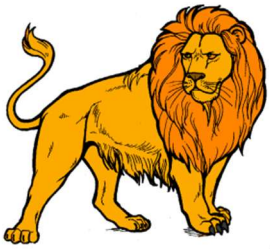
Instructions:

- Partially or fully **copied assignments** will be marked as **zero**.
- Submission will be **Online** on Google Classroom with **Screenshots** of running code in a word file.
- Also Submit .cpp file(s) separately.
- Late submissions are not allowed.
- Submission date: April 20, 2025
- Max Marks: 70

Adventure Quest is a two-player game, played on a 2-D board, in which both players try to reach the goal state (middle cell of the board) with maximum points and money. Initially, both players are given some gold and silver coins, and they will move to the next cell on their turn. Players will also try to block the passage of the opponent player by placing some hurdles, which can be bought with coins, and they will lose some points too.

When a player finds a hurdle in their path, they can either wait for some time (skip some turns) or buy some helper objects in exchange for coins. The game board also contains some gold and silver coins placed in random cells; a player can pick and place them in their box, and points of the player should update accordingly.

Money			
Gold Coins		10 Points for each coin	
Silver Coins		5 Points for each coin	
Helping Objects		**Lose points for buying**	**Use of Objects**
Sword		40 Points	When a player enters in a cell with snake or lion then, he can kill them with sword and move on. Sword can be used only twice in life of a player.

Shield		30 Points	A player can use shield to protect him against ghost attack, this will be used once.
Water		50 Points	A player can use water to protect him from fire only one time.
Key		70 Points	A player can open the door of locked cell by using key.
Hurdles		**Lose points for buying**	**Use of Objects**
Fire		50 Points	Fire up the cell of opponent and the player will be blocked there for 2 turns.
Snake		30 Points	Block the opponent for 3 turns and sent him back by 3 cells.
Ghost		20 Points	Block the opponent for single turn.
Lion		50 Points Can buy using only gold coins	Block the opponent for 4 turns.

Lock		60 Points Can buy using only silver coins	Keep the player blocked in locked cell untill he use a key.

A basic layout of Game board is presented in figure below.

Rules of the Game

1. Winner/Loser: The winner of this game is decided on the basis of two things: number of points and reaching the goal. If a player reaches the goal first but with fewer points, then the game will be a draw. A winner must reach the goal state with more points.

2. When the game starts, both players will be given the same amount of money and points. Whenever a player buys and places a hurdle in the path of the other player, the game board should be updated, and the hurdle should be visible on the board.
3. Whenever a player buys a helping object, it must be added to their box. If a player finds an object (coins) in any cell, they can put that in their box and gain points. Once a player picks an object, it will be removed from that cell, and the game board should update accordingly.

Player Movement Rules:

1. TurnBased Movement: Players take turns to move one step forward along their designated path on the board.
2. Movement Direction: Players follow a predefined path (e.g., a zigzag pattern or spiral) to reach the goal cell in the center of the board.
3. Blocked Cells: If a player encounters a hurdle (e.g., fire, snake, ghost, lion, or lock), they cannot move forward until they resolve the hurdle by:
   - Using a helper object (e.g., sword, shield, water, or key).
   - Waiting for the required number of turns (e.g., 2 turns for fire, 3 turns for a snake, etc.).
4. Picking Up Objects: If a player lands on a cell containing coins or helper objects, they can pick them up and add them to their inventory. The object is then removed from the board.
5. Skipping Turns: If a player is blocked by a hurdle and chooses not to use a helper object, they must skip the required number of turns.
6. Backtracking: Certain hurdles (e.g., snake) may force the player to move backward by a specific number of cells.
7. Reaching the Goal: The first player to reach the goal cell stops moving but must wait for the other player to finish. The winner is determined based on points (coins collected minus points lost for buying objects).
8. Dynamic Board Size: If the game progresses to the next level, the board size increases, and the players start from their respective starting positions on the new board.
9. Boundary Rules: Players cannot move outside the board boundaries. If a move would take them off the board, they must wait for their next turn.
10. Interaction with Opponent: Players cannot occupy the same cell at the same time. If a player lands on a cell occupied by the opponent, they must wait for their next turn.

Other Requirements

1. **Dynamic Board**: Your game should support a dynamic (odd) size of the board, minimum 5×5 (25 cells) to maximum 11×11 (121 cells). On completion of one game, you can move the players to the next level by increasing the size of the board or exiting the game.

2. The initial money for a 5×5 board will be 10 gold and 20 silver coins. You will pass the remaining money of both players to the next level if they want to continue the game, with some additional money (same for both).
3. **Objects**: Each time a new game is started, the number and placement of objects (gold or silver coins) will be random. You can place a minimum of 4 and a maximum of 5+5 objects on the 5×5 (25 cells) board. Similar calculations will be used for larger boards, like a minimum of 10 to a maximum of 20 objects on an 11×11 board.
4. **User Interaction**: Your game should be userinteractive and should provide proper options to the player, which they can use for every move, like picking an object from a cell, killing a snake, using an object, buying a hurdle, or buying a helping object.
5. **ObjectOriented Design**: Think about the hierarchical (inheritance) structure of your game, and design a class AdventureQuest for the final game, which will contain (Board of Items objects, and Players Objects).
    - Create a base class for game items with common functionality (e.g., name, position).
    - Derive specific item classes (e.g., coins, hurdles, helper objects) that override appropriate methods using **function overriding**.
    - Use function overriding to create unique behaviors when players interact with different items.
6. **Game State Management**: A player can stop the game in the middle and continue next time with that game or a new game. Implement a save/load system using proper copy constructors to maintain game state integrity.
    - Use **deep copy constructors** for classes that manage dynamic memory (such as the game board and player inventory).
    - Ensure proper memory management when loading and saving game states.
7. **User Interface: You will provide a graphical user interface for the game. Create a neat and clean board on the console and write the name of objects in cells with proper updates of moves. Similarly, you will update both users' boxes, money, and points information accordingly.**