ECE 250 — Winter 2024
Electrical and Computer Engineering
University of Waterloo

Due Date: Feb 12, 2024
Professor: Ziqiang Patrick Huang
Lab Instructor: Ahmed Fahmy

Lab 1:

What Time Is It, Big-Bad Wolf?

1 Overview

The objectives of Lab 1 are:

- Create the “What Time Is It, Big-Bad Wolf?” game.
- Implement and utilize a linked list class to store and update players information

2 Understanding the Game

The game “What Time is it, Big-Bad Wolf?” is a popular game played by children. In this game, one child, the Big-Bad Wolf, stands in the center, i.e. $(0, 0)$ coordinates. The other players are scattered around the wolf at their own (x, y) locations. Players take turns asking the Big-Bad Wolf, "What time is it, Big-Bad Wolf?" The wolf responds with a time (e.g., “3 o’clock”). The other players then take that many steps directly toward the wolf. At any point, the wolf can announce, “It’s lunchtime!” and catch the players closest to their location, eliminating them from the game. The game continues until all players are caught or until someone successfully reaches the wolf without being tagged.

Modified Version

We are going to consider a modified version of the game: Players may never leave the first quadrant, such that **their x and y coordinates must always be strictly positive**. A player who leaves the first quadrant during the course of the game is considered cheating, and is removed even if the wolf cannot catch them. A player who attempts to start the game outside of the first quadrant is not allowed to play. When Wolf responds “It’s lunchtime!,” all players within a distance less than 1 are caught and shall be eliminated. The game ends when specified by the user.

3 Program Requirements

The goal of this project is to write a C++ program that implements the game such that the user is able to:

1. **Spawn** players in some given coordinates, **if in the first quadrant**
2. **Update** player positions based on some time given by the user, **and take actions against cheaters**
3. **Remove** the player closest to the wolf **when its lunchtime**
4. **Print** players information that are **still in the game**.

You must **implement and use a linked list class** to store the players’ data. **Using the STL library is not allowed**. Every node in the linked list should store a player’s position as coordinates. In addition, the program has restrictions such that a player is considered cheating if they:

- Leave the first quadrant during the game.

- Attempt to start outside the first quadrant.

Any attempt to cheat will result in the removal of the player from the game. **Note:** A player is in the first quadrant **iff** both $x > 0$ and $y > 0$.

Your program must read commands from standard input (e.g. `cin`) and write to standard output (e.g. `cout`). For each input, the program is expected to execute a designated action with a specified run-time requirement and provide the corresponding output[1] message, as outlined in Table 1. **Note:** You may assume that all commands are valid.

Command	**Parameters**	**Action**	**Output**	**Run-time**
`SPAWN`	`x y` Range: $[-500, 500]$	Spawn a new player at (`x`,`y`) by adding them to the linked list, **only if** (`x`,`y`) is in first quadrant.	`success` OR `failure`	$O(1)$
`TIME`	`t` Range: $(0, 500]$	Move all players toward the wolf, according to: • `x -= t * cos(atan2(y,x))` • `y -= t * sin(atan2(y,x))`	`num of players: N`	$O(N)$
`LUNCH`	N/A	Remove all players within a distance of < 1 from the wolf.	`num of players: N`	$O(N)$
`NUM`	N/A	Output the number of children still playing the game	`num of players: N`	$O(1)$
`PRT`	`d` Range: $(0, 500]$	Print the coordinates of all players within a distance of $<$ `d` from the wolf, **if any**.	`X1 Y1 X2 Y2...` OR `no players found`	$O(N)$
`OVER`	N/A	Determine the winner: Wolf wins if no players left playing, and players win otherwise. Then, end the program	`players win` OR `wolf wins`	$O(1)$

Table 1: Input/Output Requirements

Provided Files

You are provided with the following files: `driver.cpp` where the `main()` function is implemented, `game.h` where your classes are *defined*, and `game.cpp` where your classes are *implemented*. **Do not create, delete or change the name of the provided files**. In addition, there are some examples of input files along with their corresponding output files. The test files are named `test01.in`, `test02.in`, and so on, with their respective output files named `test01.out`, `test02.out`, etc.

4 GitLab Repository

We are going to use GitLab for code management and submission. The repository is created for you with the URL: `https://git.uwaterloo.ca/ece250-w24/lab1/ece250-w24-lab1-USERID`

[1]The outputs specified in the "Output" column must be displayed as indicated. For example, in the first row, the expected output is the string "success" in lowercase.

5 Evaluation Criteria

Evaluation Focus

Output Accuracy and Performance: The primary focus of the evaluation for Lab 1 will be on the correctness of the output generated by your program and the Performance (Time complexity of the functions designed for different commands). We will use a variety of inputs and edge cases to test the accuracy of your results.

Deductions for Non-Compliance (up to -50 points): Marks will be deducted if your code does not adhere to the specified guidelines, including the prohibition of using C++ STL.

Note: It is imperative that you adhere to the prescribed utilization of linked lists as outlined in the lab manual for your assignment. Failure to comply with this directive will result in a substantial deduction of points from your overall assignment score.

Additional Considerations

Code Inspection: Following the output evaluation, we will review your code to ensure it aligns with the provided notes and guidelines.

Code Comments: Include comments to explain your code logic and design choices.

Testing: Thoroughly test your program with various inputs to ensure its correctness.

No STL: Remember that the use of any C++ STL containers or algorithms is prohibited.

Performance Evaluation: For Lab 1, we will be checking the time complexity of functions you have designed for different commands (i.e. function for SPAWN, function for TIME, LUNCH etc.)

Memory Leaks: We will be checking for memory leaks in Lab 1.

Use of AI Tools: If you utilized any AI tools in the development of your code, please include a comment in your source code indicating their use.

Code Compilation: Ensure that your code compiles without errors on ECE Linux servers (e.g., eceubuntu, ecetesla). We will make and test your code on these servers. If we cannot compile your code, you may receive a score of 0 for the marking.

Object-Oriented Design Principles:

- You must use proper Object-Oriented design principles in the implementation of your code.
- Create a design using classes that have appropriately private/protected member variables and appropriate public member functions.
- It is not acceptable to simply make all member variables public.
- You should consider separating the interface and implementation of class member functions by placing the declarations in a .h (header) file and the definitions in a .cpp (source code) file.
- Implement constructors for initialization and destructors for resource release in classes.