CSC3150 Assignment 2

1. Environment and Provided Materials

1.1 Environment

- Before starting on this assignment, make sure you have set up your VM properly. We would test all students' homework using the following environment. You can type the following command in the terminal on your VM to see if your configuration matches the test environment. If you follow the tutorials, then your VM setting should be fine, though verifying your environment again is still recommended.
- **Linux Version**

  The versions of Ubuntu from 16.04 to 22.04 are ok. You can use the following command to get it.

  ```
  csc3150@csc3150:~$ cat /etc/issue
  Ubuntu 20.04.6 LTS \n \l
  ```

- **Linux Kernel Version**

  5.4.x or 5.10.x is ok. You can use the following command to get it.

  ```
  csc3150@csc3150:~$ uname -r
  5.4.0-165-generic
  ```

- **GCC Version**

  4.9 above. Use "gcc --version" to get it.

  ```
  csc3150@csc3150:~$ gcc --version
  gcc (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0
  Copyright (C) 2019 Free Software Foundation, Inc.
  This is free software; see the source for copying conditions.  There is NO
  warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
  ```

- If your environment is not satisfied, you are still good to go, but please try to test your program with the above environment **at least once**, because you may be able to run your program on your environment but not on TAs' environment, causing inconvenience or even grade deduction.

1.2 Provided Materials

- we have provided a zip file called `CSC3150-OS-AS2.zip`, which contains:
  - `report_template`: The template for the report.
  - `source`: `hw2.cpp` is the template code provided. Please feel free to modify it. If you do the extra credit, you should add `hw2_extra_credit.cpp` to this folder.
  - `CSC3150_Assignment2_demo.mp4`: the demo video for the implemented game.

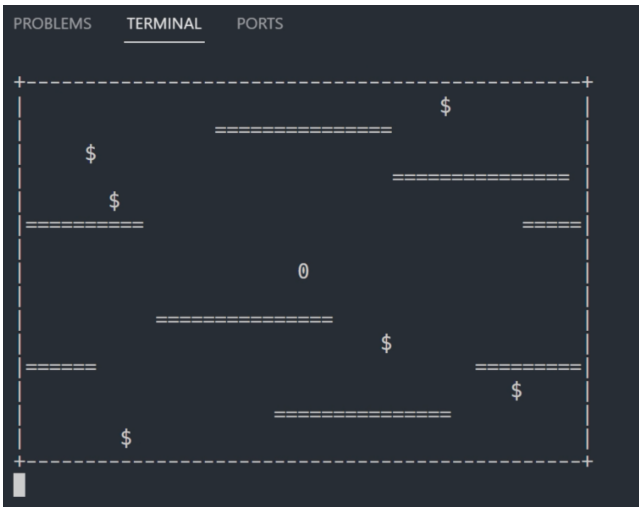
2. Task Description

Game background: The greatest wealth belongs to the bravest people! An adventurer travels alone to a dungeon, where he encounters many gold shards, but changing walls impede his progress. The adventurer's goal is to get all the gold shards without hitting the wall.

Task: In the `source` **directory** of Assignment 2, you are required to complete the multithread program to implement the game "**The Greatest Adventurer**" in `source/hw2.cpp`.

2.1 Game Rules

- In the beginning, the adventurer is in the middle of the dungeon, represented by `0`, and there are 6 walls moving towards left or right. There are 6 gold shards moving towards left or right, and the adventurer should catch all of them to win the game.



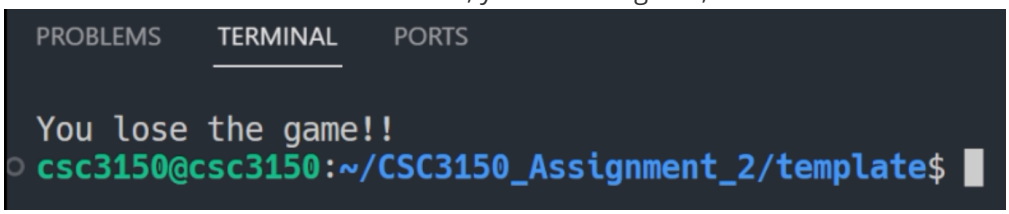
2.1.1 Basic objects

- **Details about the dungeon**
  - **representation:** several `+`, `|`, `-`
  - **size:** (including borders) height: 17; width: 49
  - To be clear, we denote rows of the dungeon with indexes 0 to 16, and columns with indexes 0 to 48 for further descriptions.
- **Details about the walls**
  - **representation:** `==========` (`'='` * 15)
  - **moving directions:** (from top to bottom) right, left, right, left, right, left
  - When the wall disappears from one border, it reappears from the opposite border
  - **initial place**: row 2, 4, 6, 10, 12, 14; random columns
- **Details about the gold shards**
  - **representation:** `$`
  - **moving directions:** random
  - **initial place**: row 1, 3, 5, 11, 13, 15; random columns
- **Details about the adventurer**
  - **representation:** `0`
  - **initial place**: row 8; column 24

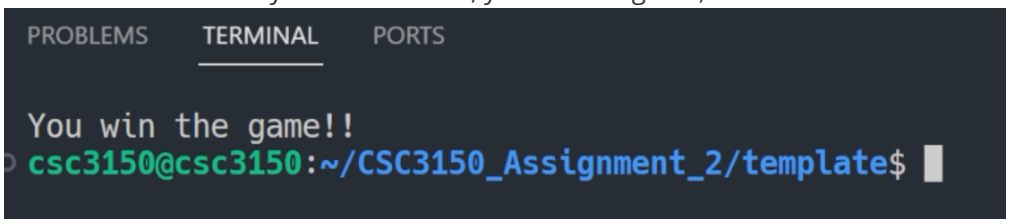
2.1.2 Game Mechanics

- when clicking `W`/ `S`/ `A`/ `D`, the adventurer will move up / down / left / right for one index.

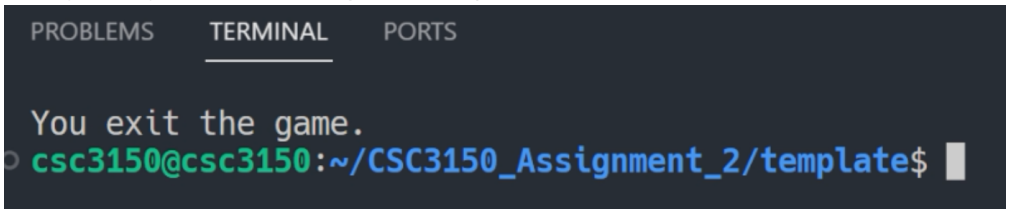
- When the adventurer touches the wall, you lose the game, and the screen should be:



- When the adventurer touches the gold shards, this shard should disappear. When all the shards are collected by the adventurer, you win the game, and the screen should be:



- At any time you click `Q`, the game will quit, and the screen should be:



2.2 Important specifications

- **Important!! You must use multiple Pthreads to implement this assignment (hw2.cpp)**, i.e., there should be **more than 3** `pthread_t` **to be created and used in your code hw2.cpp**. We recommend you assign separate threads for each object that can move.
- You should make sure your game goes smoothly and there are no possible bugs. For example:
  - The speed of the moving walls and gold shards can be specified by yourself, but it should not be too fast or too slow **(To ensure that the player can win this game, and the wall needs to traverse the dungeon within 10 seconds.)**
- There should be NO redundant characters (e.g. W, S, A, D, Q, etc) on the screen that affect the game experience. (Since you must input some characters, you need to clean them using some terminal controls introduced in the Tutorials.)
- The adventurer **CANNOT** be on or even cross the borders.
- For some of the variables mentioned above that require randomness, you need to use the `srand()` and `rand()` functions to set them.
- The template `source/hw2.cpp` has created the dungeon and the adventurer, you can follow the command in the `source/README.txt` to run the program. You can modify it and define your own functions freely.
- **If you are still confused after reading this instruction, you can refer to the demo video** `CSC3150_Assignment2_demo.mp4`**.**

2.3. Extra Credit

We have implemented this game through multi-threading, but if we **decrease the number of threads, or even use only 1-2 threads** to implement this game, what impact will it have on the game? What is the difference compared to the case with more threads?

You should implement the same game with less threads (**less than 3 threads**) in `source/hw2_extra_credit.cpp`. In your report, explain how using fewer threads will affect the game.

3. Submission and Grading Rules

3.1. Report (10 points)

You shall strictly follow the **provided latex template** for the report, where we have emphasized important parts and respective grading details. Reports based on other templates will not be graded. Feel free to add more sections and subsections, but your report should **contain the sections provided in the template**. You should mainly include the information below:

- Your name and student ID.
- How did you design your program.
- The environment of running your program. (E.g. version of OS and kernel)
- The steps to execute your program.
- Appropriate way to show that your program runs successfully. (E.g. Screenshot of your program output and some descriptions.)
- What did you learn from the tasks.

3.2. Submission

- **Due on: 23:59, March 19, 2024**
- Please note that, teaching assistants may ask you to explain the meaning of your program, to ensure that the codes are indeed written by yourself. Please also note that we would check whether your program is too similar to your fellow students' code using plagiarism detectors.
- **Late submission:** A late submission **within 15 minutes** will not induce any penalty on your grades. After that, every additional day your submission is late will **reduce your score by 10%**. (e.g., Xiao Yu submit a perfect attempt of Assignment 2 on March 21, 2024. She will get 100 * (1-0.2) = 80 points for her Assignment 2.

Here is the format guide. The project structure is illustrated below. You can also use `ls -R` command to check if your structure is fine. Structure mismatch would cause grade deduction.

```
csc3150@csc3150:~/Assignment_2_120010001$ ls -R
.:
Report.pdf  source

./source:
hw2.cpp  hw2_extra_credit.cpp
```

Please compress all files in the file structure root folder into a single zip file and **name it using your student id as the code showing below and above, for example, Assignment_2_120010001.zip.** The report should be submitted in the format of **pdf**, together with your source code. Format mismatch would cause grade deduction. Here is the sample step for compressing the root folder `xxxxx` of your whole project.

```
csc3150@csc3150:~$ ls
xxxxx
csc3150@csc3150:~$ zip -q -r xxxxx.zip . -i xxxxx
csc3150@csc3150:~$ ls
xxxxx  xxxxx.zip
```

3.3. Grading rules

Here is a sample grading scheme. Different from the points specified above, this is the general guide when TA's grading. **Please also note that we will use plagiarism detection software to check on your programs against all submissions. If you are caught plagiarizing, you will receive zero points.**

Completion	Marks
Report	10 points
Completed with good quality	80 ~ 90
Completed accurately	80 +
Fully submitted (compile successfully)	40 +
Partial submitted	0 ~ 40
No submission	0