Curtin University — Discipline of Computing

Unix & C Programming (COMP1000)

Semester 1, 2021

Assignment Two

Due: Friday, 21 May 2021, 11:59 pm

Weight: 35% of the unit

Your task for this assignment is to design, code (in C89 version of C) and test a program. This is the work extension from Assignment 1. In summary, your program will:

- Read the parameters from text file and utilize them for the game configuration.
- Add the mirror(s) on the game map as an additional game mechanic.
- Implement a generic linked list to store the map sequence.
- Write the content of the linked list to a text file on demand.

1 Code Design

Your code should be structured in a way that each file has a clear scope and goal. For example, "main.c" should only contain a main function, "map.c" should contain functions that handle and draw the map, and so on. Basically, functions should be located on reasonably appropriate files and you will link them when you write the makefile. DO NOT put everything together into one single source code file. Make sure you use the header files and header guard correctly.

Make sure you free all the memories allocated by the malloc() function. Use valgrind to detect any memory leak and fix them. Memory leaks might not break your program, but penalty will still be applied if there is any.

Please be aware of our coding standard (can be found on Blackboard under "Resources") Violation of the coding standard will result in penalty on the assignment. Keep in mind that it is possible to fail the assignment with a fully-working program that is written messily, has no clear structure, full of memory leaks, and violates many of the coding standards. The purpose of this unit is to teach you a good habit of programming.

2 Academic Integrity

You must comment your code sufficiently in a way that we understand the overall design of your program. Remember that you cannot use "//" to comment since it is C99-specific syntax. You can only use /*.....*/ syntax.

This is an assessable task, and as such there are strict rules. You must not ask for or accept help from anyone else on completing the tasks. You must not show your work to another student enrolled in this unit who might gain unfair advantage from it. These things are considered plagiarism or collusion.

Do NOT just copy some C source codes from internet resources. If you get caught, it will be considered plagiarism. If you put the reference, then that part of the code will not be marked since it is not your work.

Staff can provide assistance in helping you understand the unit material in general, but nobody is allowed to help you solve the specific problems posed in this document. The purpose of the assignment is for you to solve them on your own, so that you learn from it. Please see Curtin's Academic Integrity website for information on academic misconduct (which includes plagiarism and collusion).

The unit coordinator may require you to attend quick interview to answer questions about any piece of written work submitted in this unit. Your response(s) may be referred to as evidence in an Academic Misconduct inquiry. In addition, your assignment submission maybe analysed by systems to detect plagiarism and/or collusion.

3 Task Details

3.1 Quick Preview

Please watch the supplementary videos on the Assignment link on Blackboard. These videos demonstrates what we expect your program should do. You will expand your first assignment "Laser Tank" with more features. Therefore, we will focus on the added features in this specification. Further details will be explained on the oncoming sections.

3.2 File Input Configurations

Your executable (still called "laserTank") will accept 2 command line arguments. The first argument is the name of the input text file containing the information your program need to configure the game. The second argument is the name of the output text file (Please refer to section "Write a Log File"). The format of the input file content is as follows:

<ROW_MAP_SIZE> <COL_MAP_SIZE>

<ROW_PLAYER> <COL_PLAYER> <PLAYER_DIRECTION>

<ROW_ENEMY> <COL_ENEMY> <ENEMY_DIRECTION>

<ROW_MIRROR_1> <COL_MIRROR_1> <MIRROR_1_TYPE>

<ROW_MIRROR_2> <COL_MIRROR_2> <MIRROR_2_TYPE>

.....

<ROW_MIRROR_n> <COL_MIRROR_n> <MIRROR_n_TYPE>

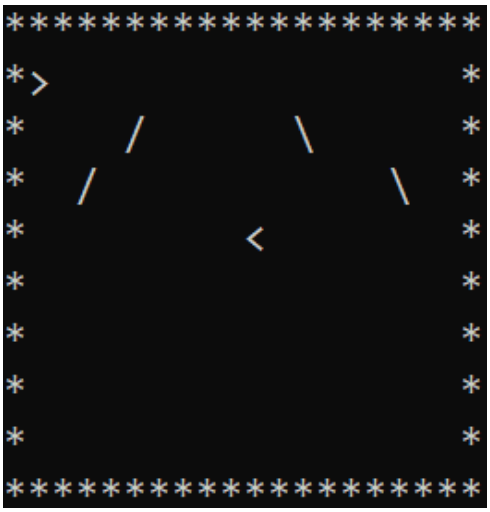
- <ROW_MAP_SIZE> and <COL_MAP_SIZE> determines the size of the playable map. You need these numbers for dynamic memory allocations of 2D char array (use malloc). The minimum and maximum are still 5 and 25 respectively.
- <ROW_PLAYER> and <COL_PLAYER> are the coordinates of the player tank. You can use these numbers as the array index of the map. Coordinate <0,0> is located at top left of the map.
- <PLAYER_DIRECTION> is the direction the player is facing upon starting the game. There are 4 choices: u, d, l, r, which is derived from the word up, down, left, right.
- <ROW_ENEMY>, <COL_ENEMY>, and <ENEMY_DIRECTION> have the same meaning as the previous explanations. However, these informations configure the enemy tank.
- <ROW_MIRROR_(number)> and <COL_MIRROR_(number)> is the location of the mirrors on the map. There is no fixed amount of the mirrors, therefore your program should be able to keep reading it until the end of the input file. (This includes the scenario where there is no mirror at all.)
- <MIRROR_(number)_TYPE> determines the type of the mirror. It is either the (b)ackslash mirror or the (f)orwardslash mirror. Refer to "Reflective Mirror" section and supplementary video for more details on the game mechanic.

This is one example:

```
10 20
1 1 r
4 10 l
2 5 f
2 12 b
3 3 f
```

3.3 Reflective Mirror

In this assignment, we add an additional item to be put on the map which is the "Mirror". This item is represented with either '/' (forwardslash) or '\' (backslash) on the map.



This add further interactions on the game mechanic:

- Player cannot move to the same location of the mirrors.
- If the player shoots the mirror, it will reflect the laser 90 degree to either left or right depending on the mirror angle (Please watch the supplementary video for example)
- The laser can bounce of multiple mirrors if it is on the laser path.
- If the laser hit the player, then the player loses the game.
- For the sake of simplicity, You can assume there will be no mirror in the line of sight of the enemy tank.

3.4 Generic Linked List

You need to complete practical 6 exercise on **generic** linked list for this assignment. This includes the necessary functions such as Linked List initialization, adding new node, freeing the linked list, etc. If you implement the non-generic linked list, you still can get some mark if it works, but you will not get full mark. Remember, just because it is from the practical exercise, it does not mean you are allowed to share this code with your classmates. Any significant collusion will be treated as Academic Misconduct, regardless of who wrote the code.

Note: If you are trying to write the generic linked list, there are 2 functions that will need a function pointer as one of the arguments: The function to write to the text file, and the function to free the memory of the whole linked list. This function pointer will process the data accordingly.

3.4.1 Data to be Stored on the Linked List

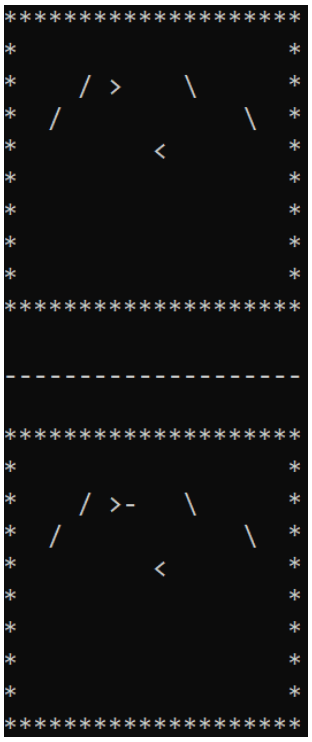
You will store a pointer to a 2D array of the map on each node of the linked list. Please make sure you make a copy of the 2D map array for each node (with malloc), otherwise all of them will point to the same array. A new Linked List node will be created to store the current copy of the map array every time the player performs an action such as moving, turning, and shooting the laser. (See the example on supplementary video).

3.4.2 Where to Create New Linked List Node

Since we want to store this map sequence in the correct order, we recommend implementing the Linked List insertLast() function. It means any new node will be stored at the end of the linked list. If you use the insertFirst() function, the order of the sequence will be reversed.

3.5 Write a Log File

You will write a function to open the output file (filename provided from the second command-line argument) and then print the content of the linked list to it, separated by a line for each printed map. You can use the default font color for the laser on the output text file. You can see the example as followed:



3.5.1 When to Call this Function

There are 2 occasions when you need to call this function you create:

- When the game is over (either losing or winning). In this case, the output file will contain the map sequence from the beginning of the game until the end.
- When the player enter the command 'l' (l for log) on the action prompt. You can add an additional line on the game instruction such as "l to save the Log". When this occurs, you have to write whatever content the linked list contains at the time to the output file correctly. Even if the player close the game by force afterwards (e.g closing the terminal OR pressing CTRL+C), the output file should have been written properly. Keep in mind that the player might use this command multiple times in the middle of the game, therefore the output file should have the most recent log information.

Note: You can open and close the same file for writing purpose every time you call the function. Opening the file with "w" mode will ensure the file is overwritten every time. Closing the file will ensure the output stream is flushed and written to the file.

3.6 Makefile

Like in assignment 1, you should write the makefile according to the format explained in the lecture and practical. It should include the Make variables, appropriate flags, appropriate targets, and clean rule. We will compile your program through the makefile. If the makefile is missing, we will assume the program cannot be compiled, and you will lose marks.

3.7 Assumptions

For simplification purpose, these are assumptions you can use for this assignments:

- The location of all the objects (tanks and mirrors) from the input text file will not collide with each other. But, you still need to check if the objects are outside the boundary of the map.
- There will be no mirror in front of the enemy tank. So, the enemy's line of sight is clear.
- The player tank will NOT be in front of the enemy tank immediately upon starting the game. (NO instant lose)
- Any data read from the text file is always on correct datatype and in lower case (for letters).

4 Marking Criteria

You have to submit a valid attempt for this assignment. If your submission is missing most of the functionalities specified above, the Unit Coordinator might regard it as insufficient attempt and nullify the mark. This is the marking distribution for your guidance:

- If your code is using C99/C11 syntax, we will take 10 marks off. (-10 marks)
- Properly structured makefile. (5 marks)
- Program can be compiled with the makefile and executed successfully without immediate crashing (5 marks)
- Usage of header guards and sufficient in-code commenting (5 marks)
- The whole program is readable and has reasonable framework. Multiple files are utilized with reasonable category (5 marks). If you only have one c file for the whole assignment, you will get zero mark on this category. If you #include the .c files instead of .h files, you get zero mark as well.
- Avoiding bad coding standard. (10 marks) Please refer to the coding standard on Blackboard. Some of the most common mistakes are:
  - Using global variables
  - Calling exit() function to end the program abruptly
  - Using "break'' not on the switch case statement
  - Using "continue''
  - Having multiple returns on a single function.
- No memory leaks (10 marks) Please use valgrind to check for any leaks. Keep in mind that freeing linked list and the data correctly can be quite tricky. If your program is very sparse OR does not use any malloc(), you will get zero mark on this category. If you don't have linked list implemented, you get zero mark on this category.
- Functionalities:
  - Able to read map configuration information from the text file (file name given in command line argument) and utilized correctly to configure the game. If the map size is outside the reasonable range OR any of the game object is located outside the map range, the game should not start. (10 marks)
  - Mirrors are shown correctly on the map. Laser should change orientation when it hits the mirror (From '-' to '|' OR Vice Versa). (5 marks)
  - Implementation of the laser animation is implemented correctly when it hits the mirror. The change of the direction is correct depending on the mirror type and initial laser direction. Hitting multiple mirrors will change the laser path multiple times. The laser will keep moving until it hits either the enemy tank, player tank, or the map border. (15 marks)
  - Implement a linked list to store the 2D map array on every player action. The map sequence should be stored in the correct order (from first move until the last move). (10 marks)

- The implemented linked list is the generic linked list. It means the datatype of the data for each node should be ”void *”. It also means that the function to ”write to text file” and ”freeing the linked list” should have a function pointer as one of the arguments to process the data. (10 marks)
- Implement the function to write the content of the linked list to the output file in the right format. This function can be either called manually through player action prompt ’l’, or automatically called when the game ends. (10 marks)

5 Short Report

If your program is incomplete/imperfect, please write a brief report explaining what you have done on the assignment. This report is not marked, but it will help us a lot to mark your assignment. Please ensure your report reflects your work correctly (no exaggeration). Dishonest report will lead to academic misconduct.

6 Final Check and Submission

After you complete your assignment, please make sure it can be compiled and run on our Linux lab environment. If you do your assignment on other environments (e.g on Windows operating system), then it is your responsibility to ensure that it works on the lab environment. In the case of incompatibility, it will be considered not working and some penalties will apply. You have to submit a **SINGLE** tarball (see practical worksheet for the tarball creation instruction) containing:

- **Declaration of Originality Form** – Please fill this form digitally and submit it. You will get zero mark if you forget to submit this form.
- **Your essential assignment files** – Submit all the .c & .h files and your makefile. Please do not submit the executable and object (.o) files, as we will re-compile them anyway.
- **Brief Report** (if applicable) - If you want to write the brief report about what you have done on the assignment, please save it as a PDF or TXT file.

The name of the tarball should be in the format of:

<student-ID>_<your-full-name>_Assignment2.tar.gz
Example: 12345678_Antoni-Liang_Assignment2.tar.gz

Please make sure your submission is complete and not corrupted. You can re-download the submission and check if you can compile and run the program again. Corrupted submission will receive instant zero. You can submit it multiple times, but only your latest submission will be marked. Please make sure your latest submission is a complete submission, and not just a small file to complement the earlier submission. If your latest submission is a late submission, we will treat it as late submission regardless of your earlier submission is on time or not.

End of Assignment