Worm Game Project Report

I. Game rules

The game is inspired by Nokia's snake game. There are 3 levels with 3 different maps from easy to hard correspondingly. In each map, there is a door covered by walls. The door is the only way leading to the next level. The player controls the worm by arrow keys to collect foods. After a certain amount of foods collected, the worm can break the wall for once. To break another wall, the worm has to collect more food. In this game, you never win, you can just live longer or shorter. The game ends when you run into a wall or bite yourself.

To compile the code, enter command: gcc --std=gnu99 worm.c -o worm.exe

II. Result

In this project, many external libraries are employed such as time.h to use random function rand(), sys/select.h to use select() function to set time out for reading arrow keys, stdbool.h to use Boolean data type, unistd.h to change between raw and cooked mode for console, string.h to manipulate strings.

During the implementation of this project, the most difficult problem faced is the lack of debugging ability which leads a lot of time consumed to print variables to the console at different stages of the program to debug. Additionally, the mode changing part is also time consuming.

After this project, I learned about the escape codes in Linux based terminal. Escape codes enable me to change default settings such as color, cursor position, and many other interesting features. Also, I found out enum which makes my code more readable.

Generally, I think my code is at intermediate level, but if I have to point out my favorite functions, I think the DrawWalls() and GenerateFood() functions are preferable in which the walls are drawn based on level and changable borders of the map and generate foods based on each map. It means that the walls adapt to the change of map borders. However, the program still needs a lot more development. Some of my functions namely AmIDead() takes too many arguments, and I am not sure it is a good idea to do so.

III. Self Evaluation

Personally, I think I am good at logical thinking, problem solving, and debugging. Also, I think should improve on using standard libraries instead of creating my own functions. Getting familiar with best practices in C programming is also a need for me.

IV. Flowchart