# CSE 379 Lab 7: Pac-Man

Daniel Jin (djin3)
Motaz Khalifa (motazkha)

---

## 1) Purpose of the Program

### a) Objective

The objective of this program is to play the classic arcade game, Pac-Man. The user will control Pac-Man using the w, a, s, and d keys to move up, left, down, and right respectively. The objective of the game is to eat all pellets on the board while trying to escape from the four ghosts. There are power pellets available that, when eaten, grant Pac-Man the ability to eat the ghosts. Upon eating all pellets on the board, a new level is reached, a new board is loaded, and the speed of the game is increased. The game can be paused by pressing “p.”

### b) Debugging Steps

Our implementation requires a lot of flags that we chose to store into memory. Whenever we encountered something wrong, we went into debug mode in Code Composer Studio, and looked through every line and used the memory brower. To avoid as much debugging as possible, we broke down this assignment piece by piece, and started by laying down the foundation to our implementation; we started with elements of the game that we knew we could get done. For example, we focused on getting Pac-Man moving first since we knew how to do this from the previous lab.

### c) How To Run The Program

Open PuTTy using the same process as previous labs, but this time, set the baud rate to 500000. In Code Composure Studios, click Debug → Resume. You will see — in PuTTy —an introduction to Pac-Man. To play the game, follow the instructions. The keys to move are w, a, s, and d. The user will be able to pause by pressing “p” and can quit the game from the pause menu. The rules for this game are similar to the classic arcade game.

### d) References

The Tiva™ TM4C123GH6PM Microcontroller Data Sheet was referenced for information about the necessary registers for this lab.

## 2) Division of Work

Daniel wrote the initialization of the necessary elements that were utilized in this lab. Daniel initialized UART0, both timers, and the RGB LEDs. Daniel also worked on all the elements of Pac-Man's movement. Motaz worked on all the elements of the ghosts' movement. Motaz also handled the leveling up, whenever a power pellet is eaten, and whenever a ghost or Pac-Man is eaten. The score, pause function, and speed of the game were worked on together.

## 3) Logic

### a) Summary of Main Program

This subroutine first initializes UART0, the timers, and the interrupts, and runs initialize_board. The subroutine then goes into an infinite loop. Every interrupt, either by the keyboard or either timer, is handled inside an interrupt handler.

### b) Summary of UART0_Handler

This subroutine looks at the game status flag, and acts accordingly to the flag. Depending on the flag and what is inputted, this handler will either launch the game, go to and from the pause menu, and update the direction that the user wants Pac-Man to move.

### c) Summary of TimerSlow_Handler

This subroutine looks at the game status flag and acts accordingly to the flag. If a power pellet has been eaten, then a 8 second timer counts down and this subroutine moves the afraid ghosts. If a power pellet isn't in effect, then this subroutine moves Pac-Man.

### d) Summary of TimerFast_Handler

This subroutine looks at the game status flag and acts accordingly to the flag. If a power pellet hasn't been eaten, then this subroutine moves the hostile ghosts in accordance to the rules of the games. If a power pellet has been eaten, then this subroutine moves Pac-Man.

The TimerSlow_Handler represents the slower speed of the two speeds, and TimerFast_Handler represents the faster speed of the two speeds. If a power pellet hasn't been eaten, then Pac-Man is moved by the slower timer or the TimerSlow_Handler. If a power pellet hasn't been eaten, then the speeds of Pac-Man and the ghosts are swapped, making TimerSlow_Handler control the afraid ghosts.

## e) Summary of Intermediate Subroutines

*i) initialize_board*

This subroutine initializes everything that is subject to change in the game, such as level, score, locations, flags, etc.

*ii) perform_switch*

This subroutine moves ghosts around the board while maintaining the character in its previous location, i.e. if a ghost is on an empty space and moves over, the empty space should remain an empty space.

*iii) throw_in_box*

This subroutine moves a designated ghost to the center box and updates the imprisoned flag in the ghost to 1 to reflect its imprisonment.

*iv) move_pacman*

This subroutine lights up the RGB LED accordingly, and if all pellets are eaten, this subroutine immediately launches the next level. This subroutine handles all of Pac-Man's movements and scenarios that it can encounter; this subroutine acts accordingly if Pac-Man runs into a pellet, an empty space, a hostile ghost, an afraid ghost, etc.

*v) start_new_level*

This subroutine resets the board and the locations of Pac-Man and the ghosts. The level is increased and the speeds of the game are increased by 25%.

*vi) print_pause*

This subroutine pauses the board and outputs instructions to restart, quit, or resume.

*vii) illuminate_RGB_LED*

This subroutine changes the color of the RGB LED depending on the number of lives and the power pellet status.

*viii) lose_life*

This subroutine decrements the amount of lives. If there would be no more lives, then the screen launches the game over prompt. If there are more lives, the locations of Pac-Man and the ghosts are reset.

*ix) move_ghosts_hostile, move_ghosts_afraid*

This subroutine sets up flags before ghosts are actually moved across the board. These flags include the previous location, the direction, and the character that the ghost is leaving behind (if a ghost is on an empty space, for example, and the ghost moves away, the empty space should remain an empty space).

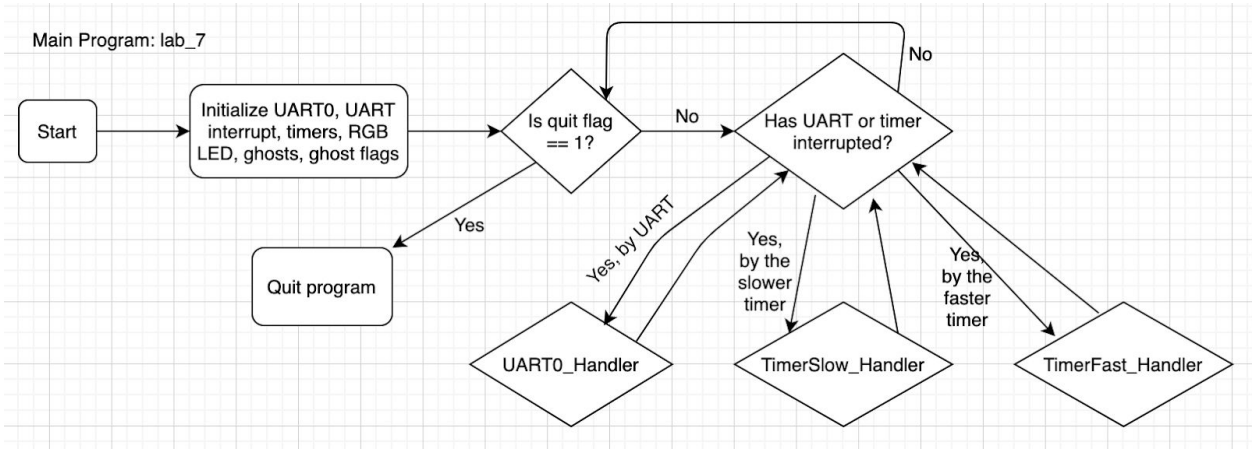
In move_ghosts_afraid, the character for ghosts are changed from “M” to “W” to reflect the power pellet effect.

*x) move_ghost_hostile, move_ghost_afraid*

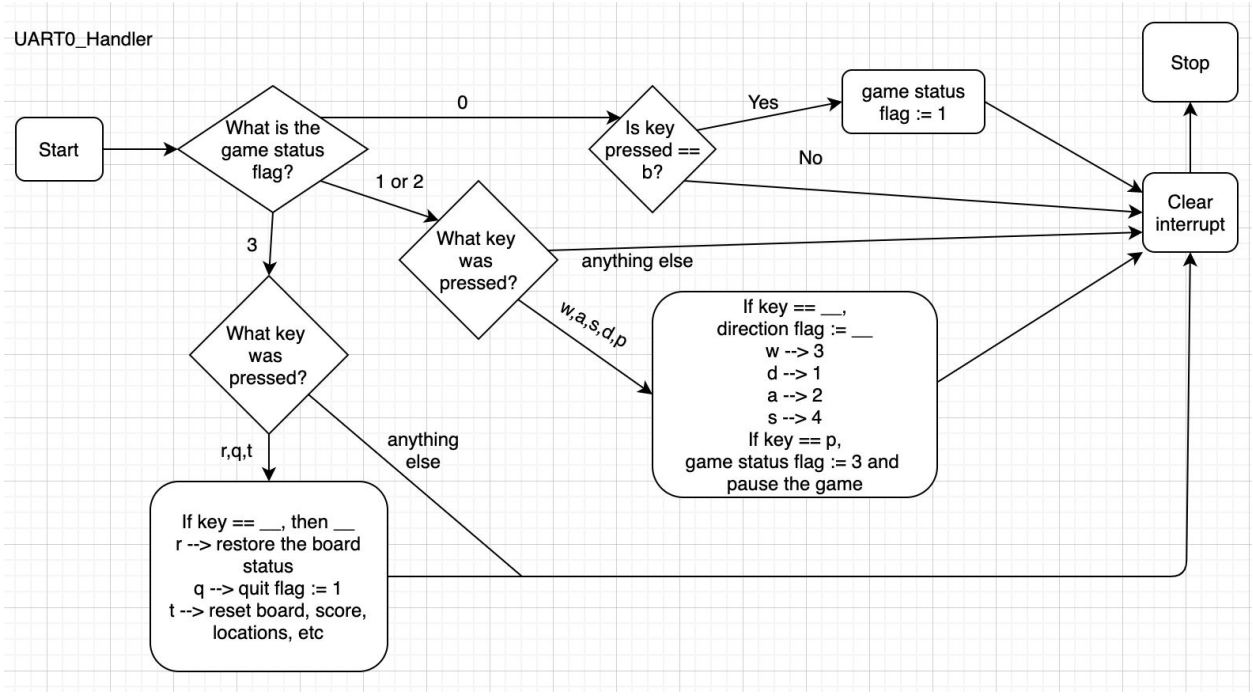
This subroutine moves ghosts across the board. The movement of the ghosts reflect the rules of the game: if ghosts are in the line of sight of Pac-Man and four spaces away, then the ghosts will go towards (if ghosts are hostile; if a power pellet hadn’t been eaten) or away from (if ghosts are afraid; if a power pellet had been eaten) Pac-Man.

In move_ghost_afraid, if a ghost meets Pac-Man, the ghost is then moved into the center box.

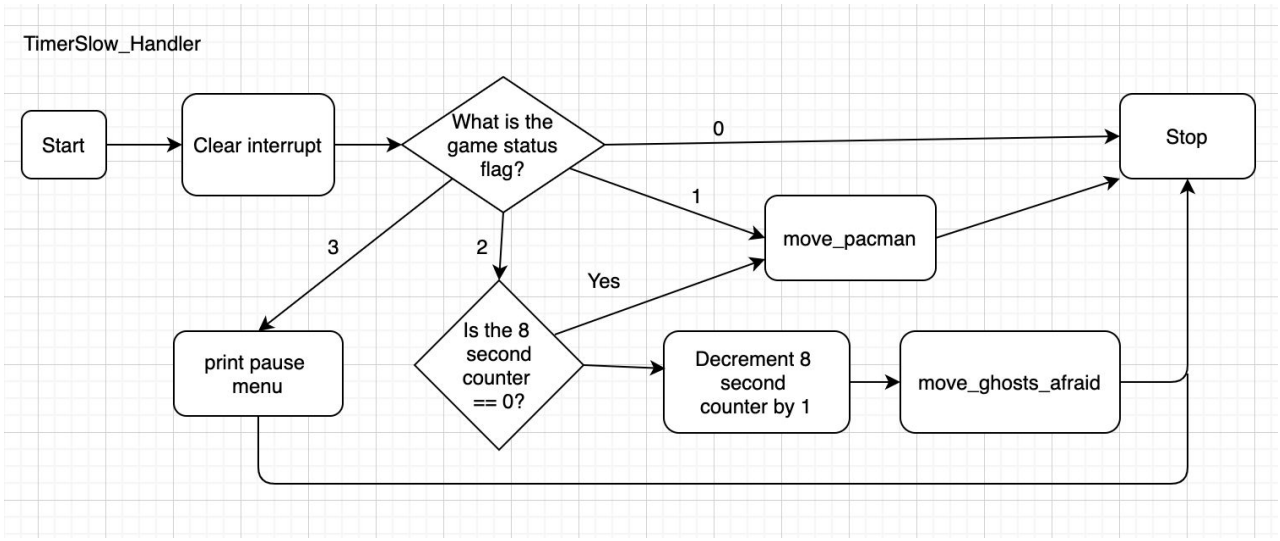
## f) Flowchart of Main Program



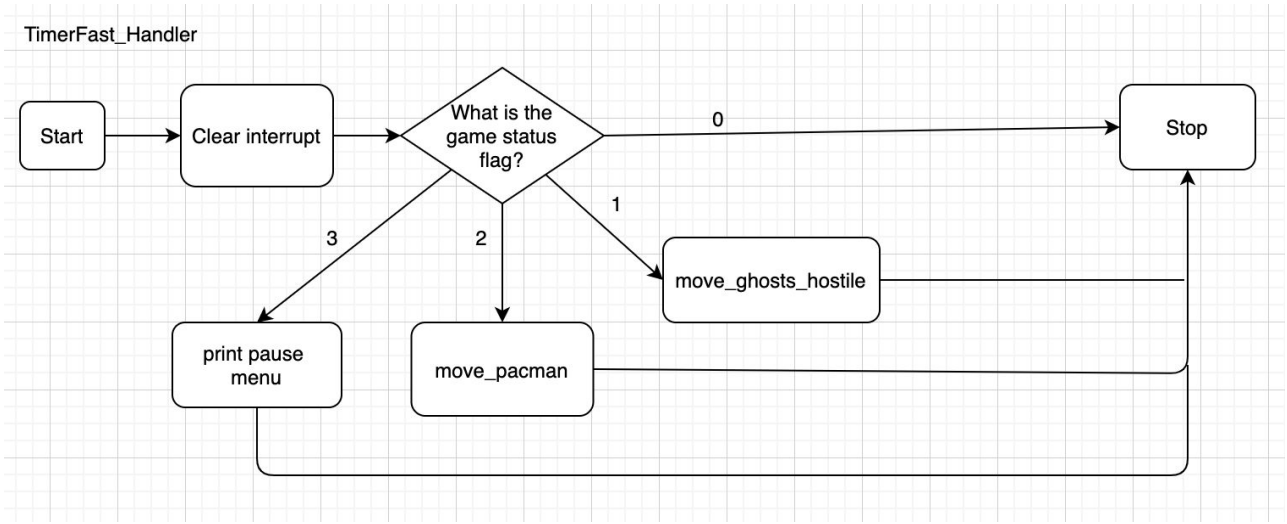
## g) Flowchart of UART0_Handler




## h) Flowchart of TimerSlow_Handler

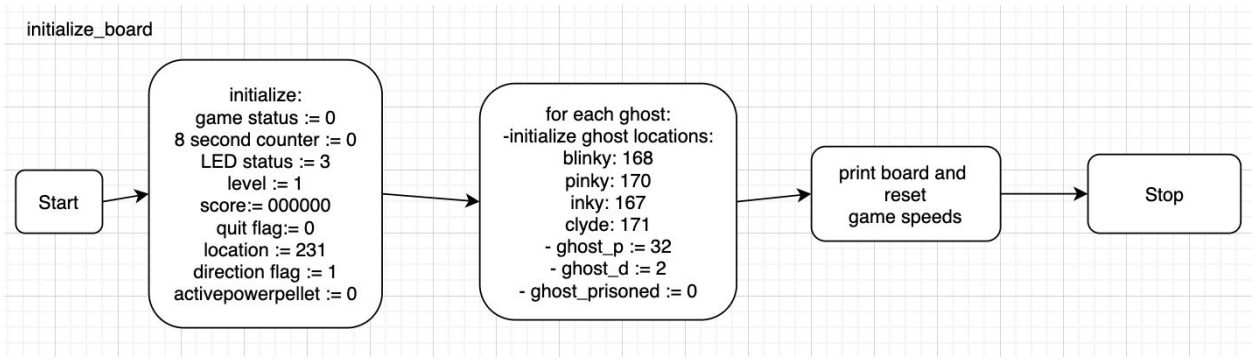


## i) Flowchart of TimerFast_Handler

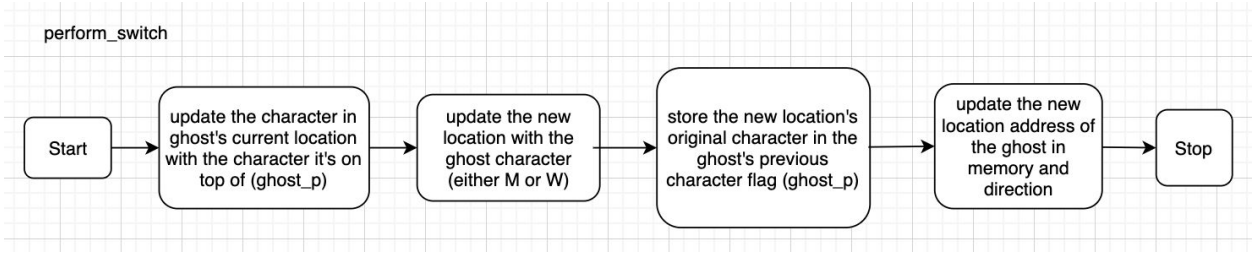



## j) Flowchart of Intermediate Subroutines

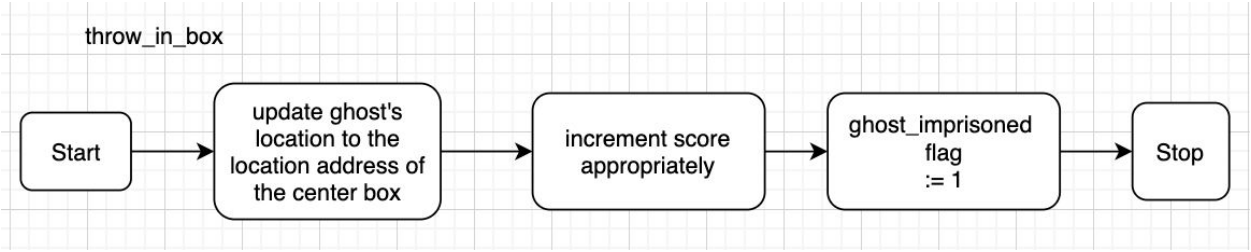
*i) initialize_board*



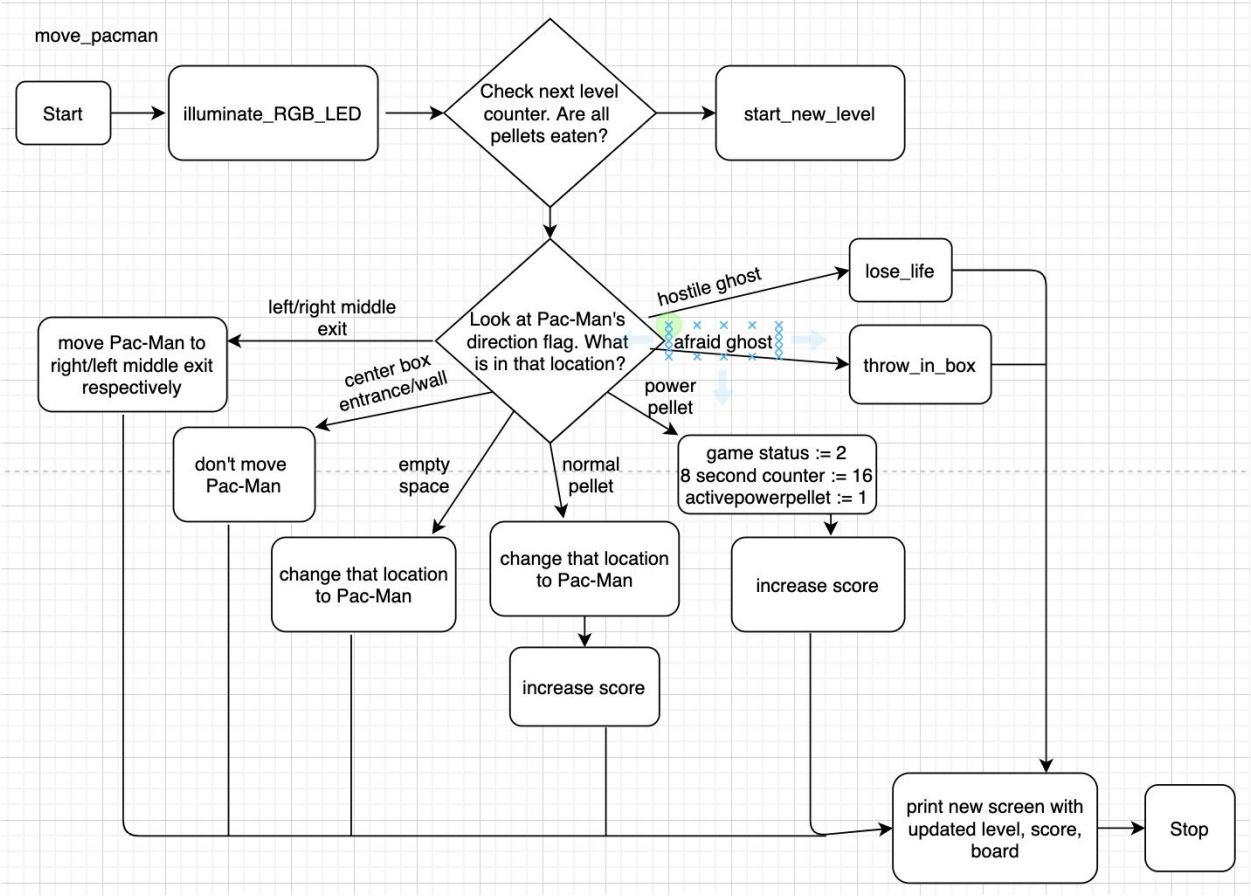

*ii) perform_switch*



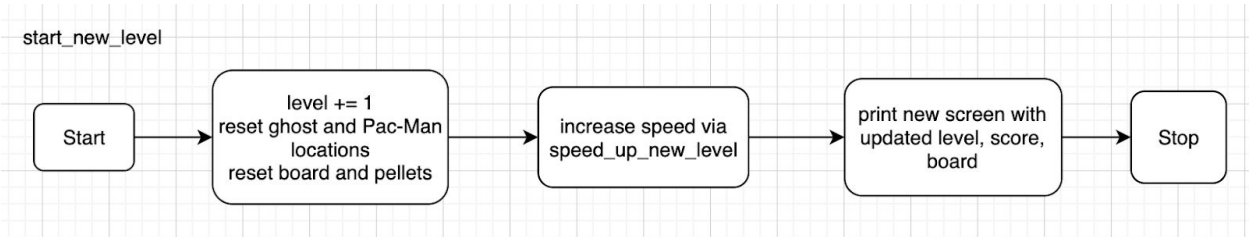
*iii) throw_in_box*



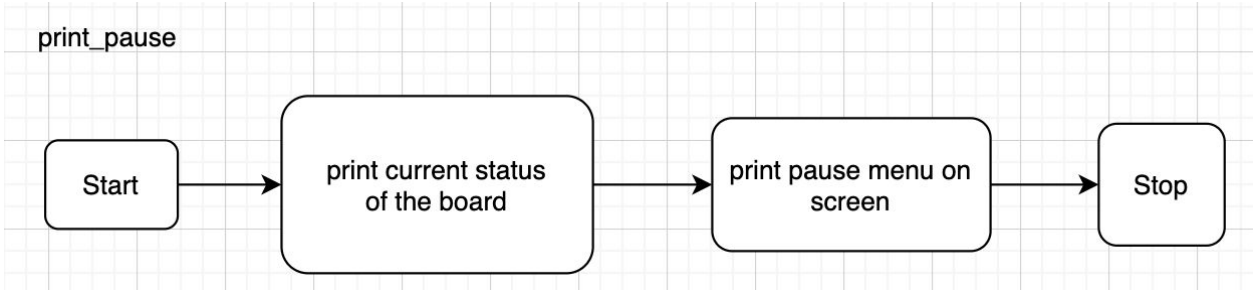

*iv) move_pacman*



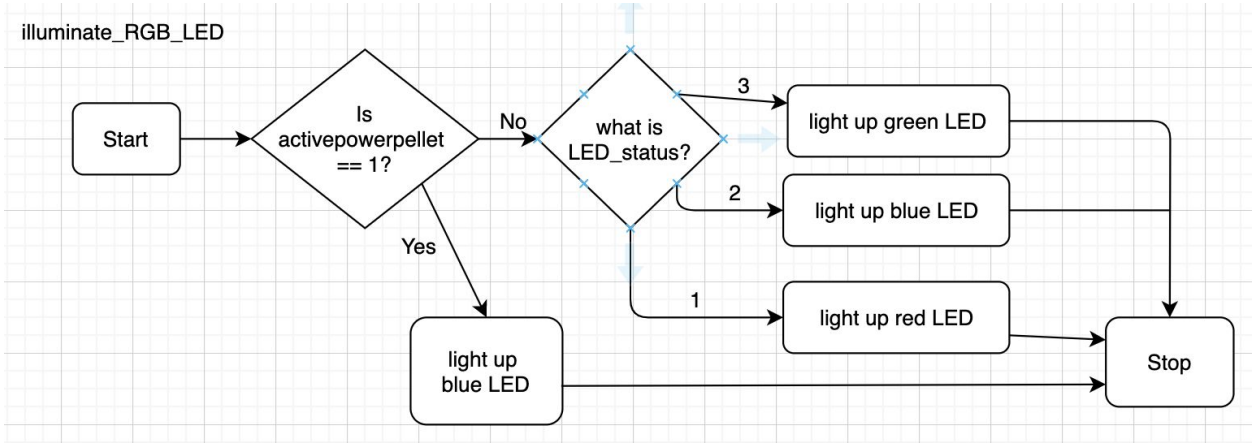

*v) start_new_level*



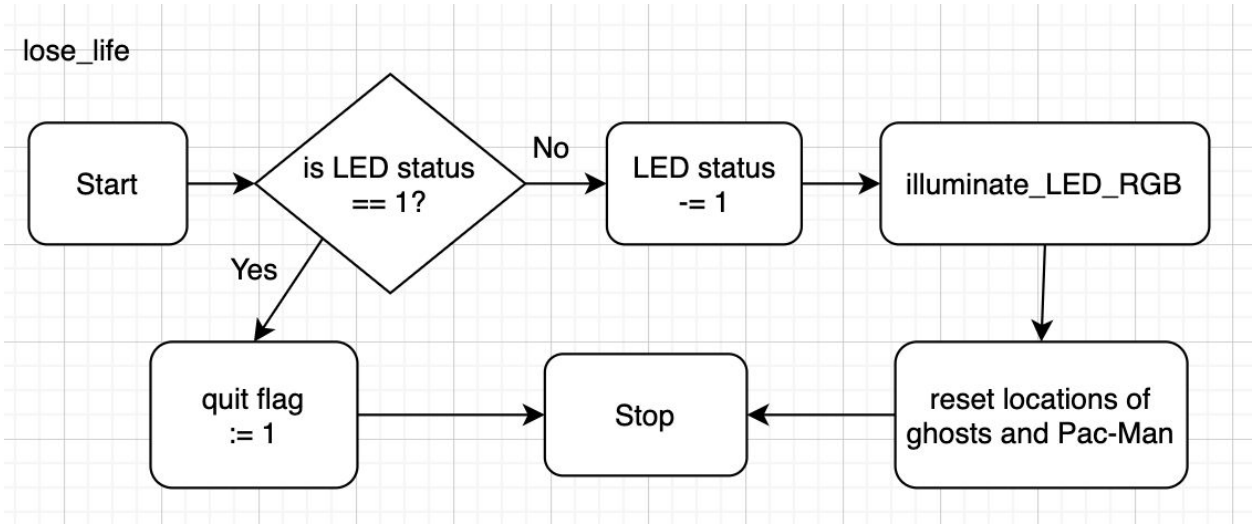
*vi) print_pause*



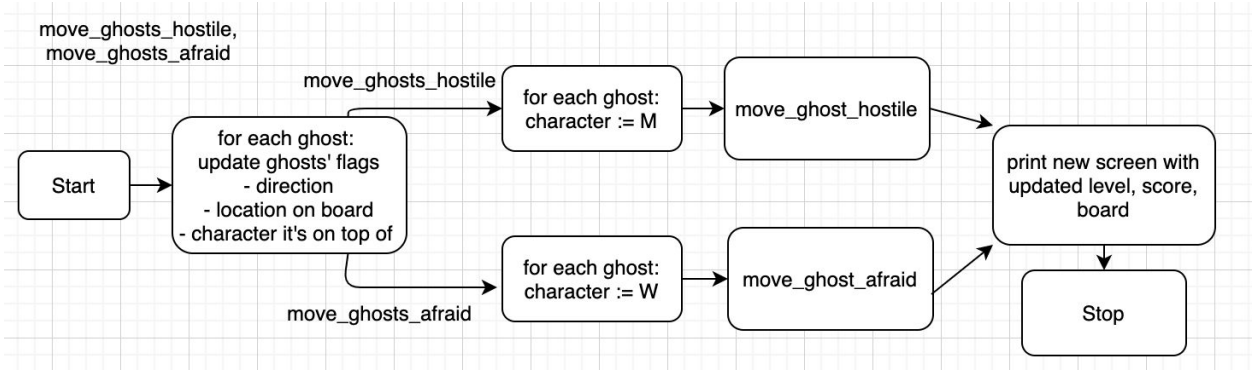

*vii) illuminate_RGB_LED*

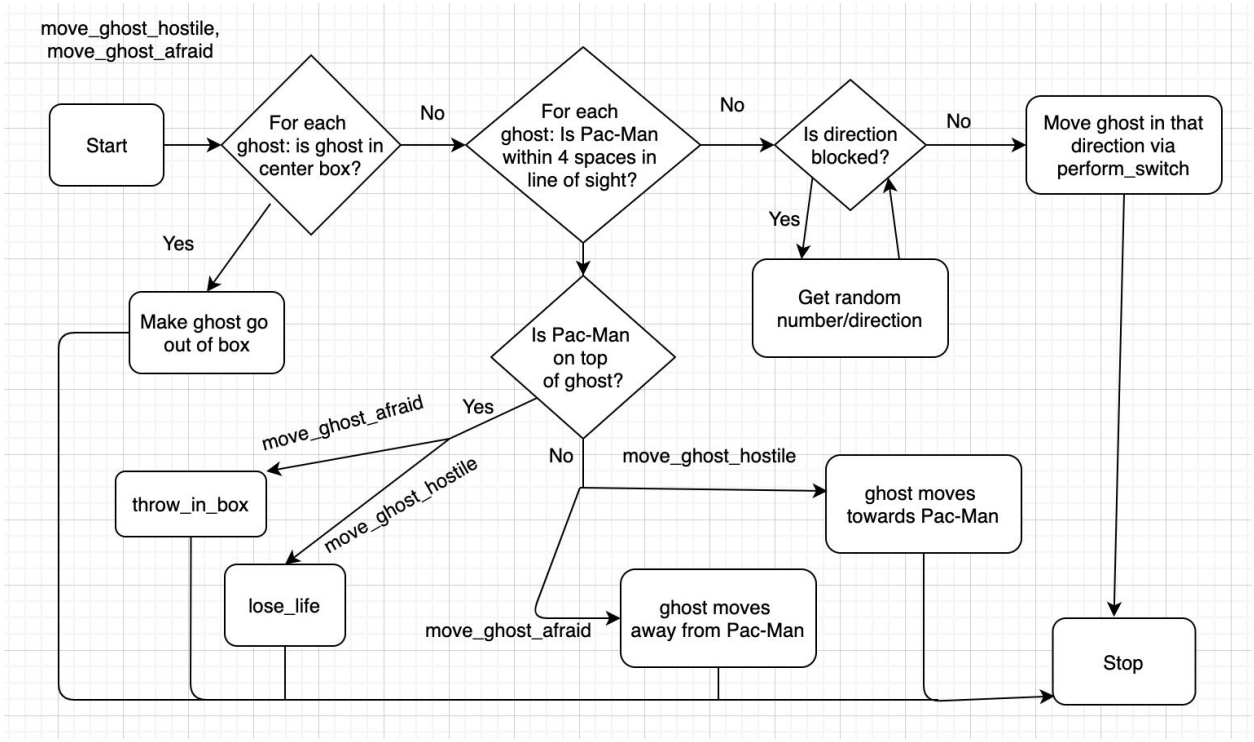



*viii) lose_life*



*ix) move_ghosts_hostile, move_ghosts_afraid*




*x) move_ghost_hostile, move_ghost_afraid*