Department of Mathematics and Computer Science
University of the Philippines Baguio

MACHINE PROJECT 1: Snake Game
(CMSC 11 - Introduction to Computer Sciences)

General Requirement. Design and implement a console-based Snake Game in the C programming language played on a 15×15 grid.

Snake Game Mechanics

Navigation. To control the snake, the characters `W`, `S`, `A` and `D` are used. `W` is used to make the snake move UP, `S` to move DOWN, `A` to go LEFT and `D` to move RIGHT. When the snake's head collide with any part of itself, the borders (`#`) or the blocks (`B`), the game ends and the user loses.

Modes. The user will have a choice to play two modes:

- ▷ `EASY` mode − The grid will have no blocks.
- ▷ `DIFFICULT` mode − The grid will include blocks on the board.

Collecting Food. There will be food randomly distributed across the board indicated by `F` (or any symbol you wish). The snake will increase its length by 1 as it collects the food and the player will earn points (the number of points will be the programmer's discretion). Displaying the points will either be during the game or after the player collects all the food and exits the board.

Board Exit. How will the game end with the user winning? The game will end if the snake moves out at a *portal*, a $-marked part of the board after all the food have been eaten. Once the last part of the tail of the snake is no longer visible, the game is done and the player wins.

Implementation

The program must be implemented using an array and functions. It is the programmer's discretion on the number of functions to utilize. For example, you can create a function that will display the menu, increase the snake's size when it eats, or whenever you want to update the board.

Program Specifications

1. The program will start with the following minimum menu:

```
The Snake Game:
Choose an option:
(1) Start Game
(2) Instructions
(3) Exit
```

Upon choosing Option `(1)`,

- The user will be asked the Game mode: `Easy` or `Difficult`. If the user chooses `Difficult` mode, ask the user to input the number of blocks to be placed randomly. The number of blocks must be between 5 and 15, and must not be placed at the boundary.
- Then, the game asks the user to input the number of food to be eaten. The number of food must be between 1 and 10.
- The board will appear (a 15 x 15 two-dimensional array). See sample grid below:

```
#  #  #  #  #  #  #  #  #  #  #  #  #  #  #
#  @  F                                    #
#     F                                    #
#     F                                    #
#     F  F           B        B            #
#     B                             F      #
#                    F                     #
#        F                    F     B      #
#                                          #
#                                          #
#     B                                    #
#                                          #
#           B                    B  B  F    #
#                                          #
#  #  #  #  #  #  #  #  #  #  #  #  #  #  #
```

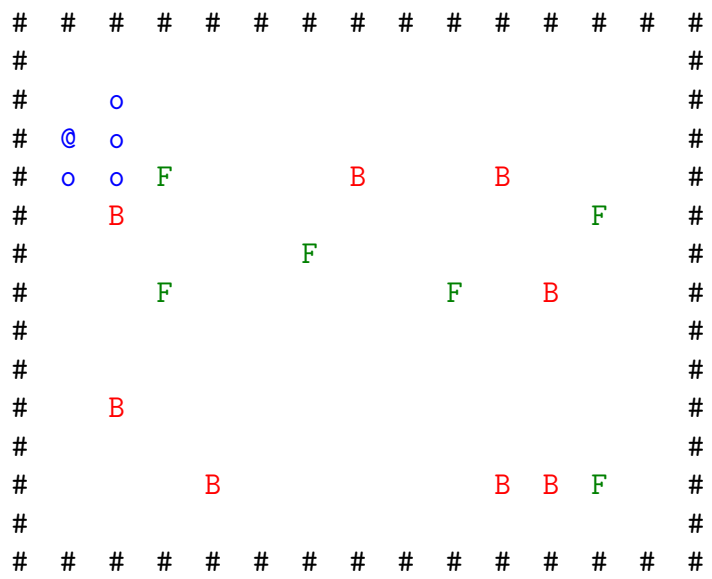
2. The snake will start at the coordinates `(1,1)` *(top-left of the board)*, length 1 (head only). Use the symbol '`@`' for the snake's head and the symbol '`o`' for the snake's body. The snake does not move automatically. The user inputs a direction at each step:

   `W- Up,  S- Down,  A- Left,  D- Right`

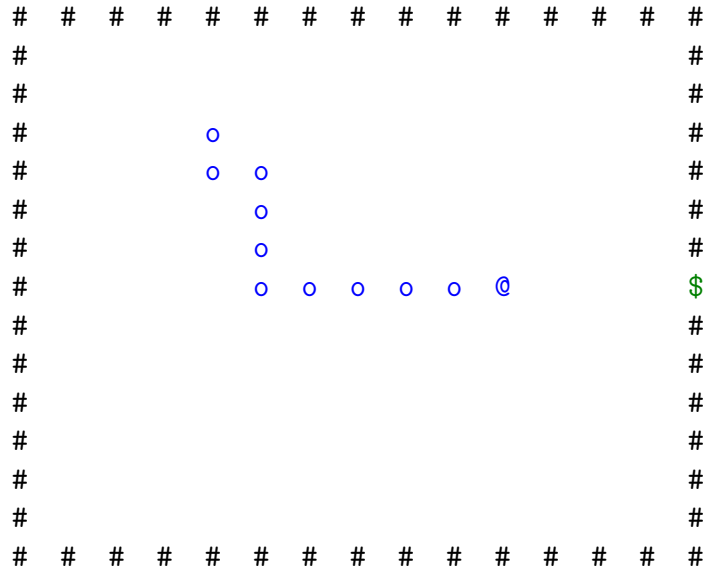
   The snake grows by one segment each time it eats food. Specifically, a segment is added **after** a food is eaten.

3. Hitting the blocks will end the game. Below are some illustrations:

   - At the initial configuration, if `A` or `W` is pressed, player loses when it collides with the boundaries and not all food has been eaten.
   - In the same configuration as above, after pressing `DSSSAWD` *(consecutively)*, the snake would have eaten itself and the game ends. See grid below.



4. After all the foods have been eaten, the blocks (if there are any) will disappear from the grid, and the $-portal (exit door) appears at the coordinate `(7,14)` *(middle-right boundary of the board)*. The snake will have to reach the portal and the last of its tail disappear before the player wins.



In the illustration, after pressing D 14 times, the snake would have disappeared from the board and the player wins.

```
# # # # # # # # # # # # # # #
#                           #
#                           #
#                           #
#                           #
#                           #
#                           #
#                           #
#                           $
#                           #
#                           #
#                           #
#                           #
#                           #
#                           #
#                           #
# # # # # # # # # # # # # # #

 You win!
```

(Optional) As the game is played, there can be an option to return to the main menu (example: user presses Q). You may use `graphics.h` for better user interface. An additional 10% (maximum) will be given if the functions of `graphics.h` are used in the project.

5. Once the game ends (or is terminated), the program must ask the user if they want to play the game again:

```
 Game over! Do you want to play again?
(1) Yes
(2) Exit
```

If user chooses `Yes`, the main menu is displayed. Otherwise, the game terminates.

Project Guidelines

- Students must work in groups of 2-3 members.
- All files must be in a zip file: `CS11_MP1_<lastname/s>.zip`. For the working file of your code, follow the filename format:

  `CS11_MP1_<lastname/s>.c.`

  Check carefully if your submitted code is running. Codes with syntax errors will be automatically given a score of 0.
- Screenshots of the output of each part of the menu saved in IMAGES folder
- The project documentation includes:
  (a) Introduction − brief description of the program
  (b) Description of data structure/s (major variables) used
  (c) Algorithm (explained using a pseudocode or flowchart)
  (d) Error handling
  (e) Application manual (how to use your program)
  (f) Contribution statement section − each member briefly describes their specific contributions to the project
- Errors in user input must be handled such as if user presses another key other than the required keys, etc.
- **Deadline of Submission. 10 November 2025**. Late submission will incur a deduction of 5% per day (max of 8 days late), while early submission will incur 2% additional per day (maximum of 5 days).

---

aalumague.kmgonzales.ay25-26.sem1