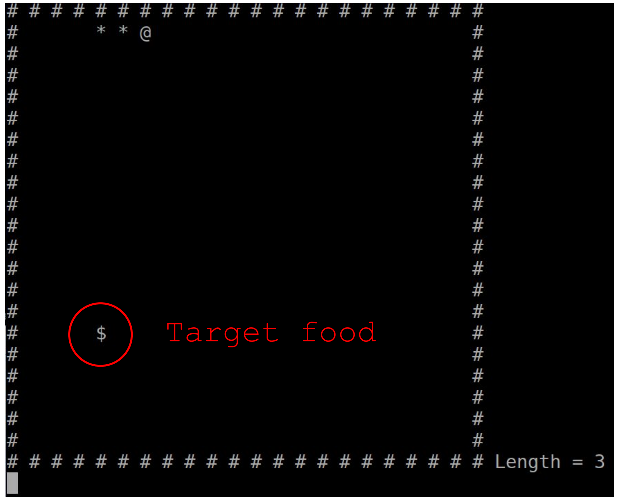
Project 2

Snake Game

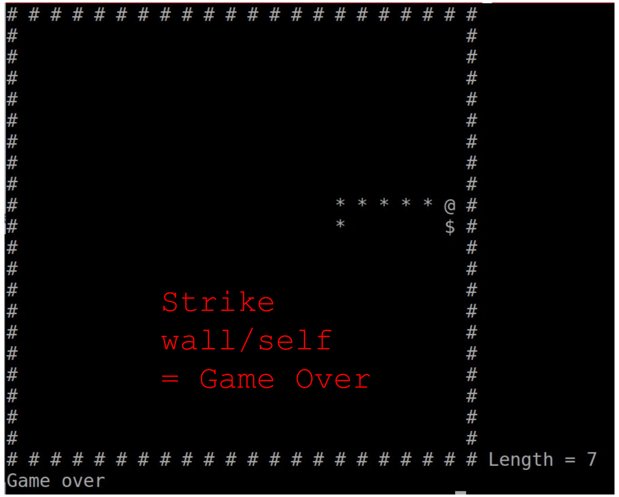
Guidebook

Game Instructions

1) Use arrow keys to command the movement of snake
2) When snake 'eats' a food, it will grow in length and another food will randomly appear on the map
3) The game ends when the snake hits a wall or its own body



```
# # # # # # # # # # # # # # # # # # # # # #
#       * * @                             #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#       $                                 #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
# # # # # # # # # # # # # # # # # # # # # # Length = 3
```



```
# # # # # # # # # # # # # # # # # # # # # #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                             * * * * * @ #
#                             *         $ #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
# # # # # # # # # # # # # # # # # # # # # # Length = 7
Game over
```

How can we represent the snake and the map?

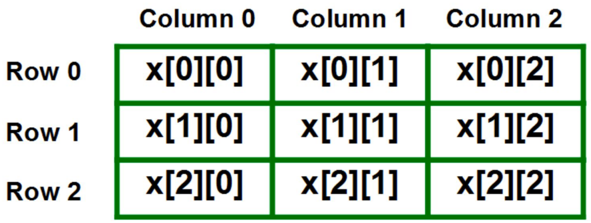
Represent the map with a 2D integer array, map[H][W]

If value in the cell is 0, cast it as blank

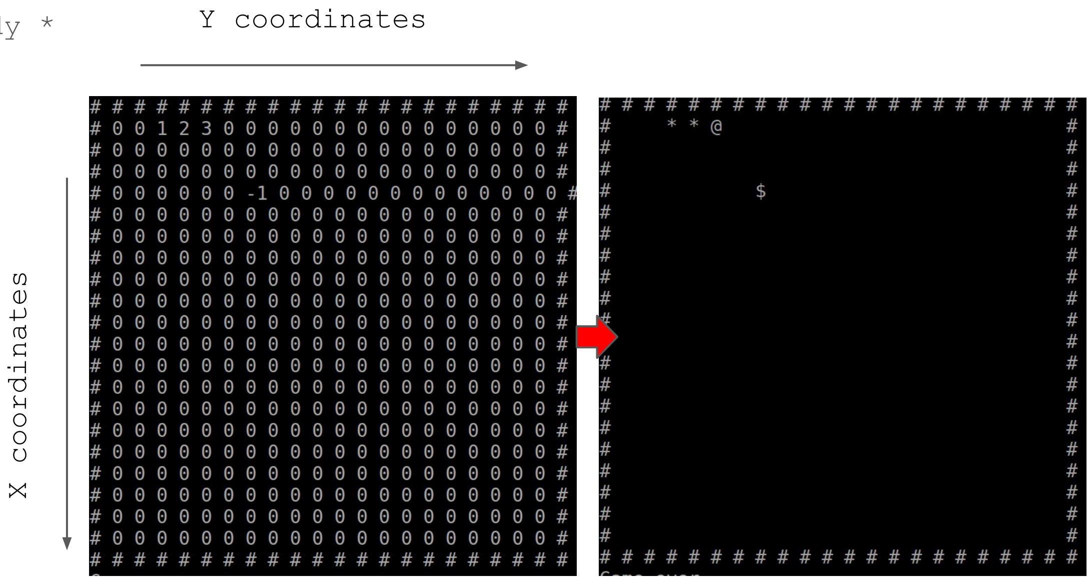
If value in cell is snake head, cast as @

For other +ve values, cast it as snake body *

If value in cell is -1, cast it as food $



	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]



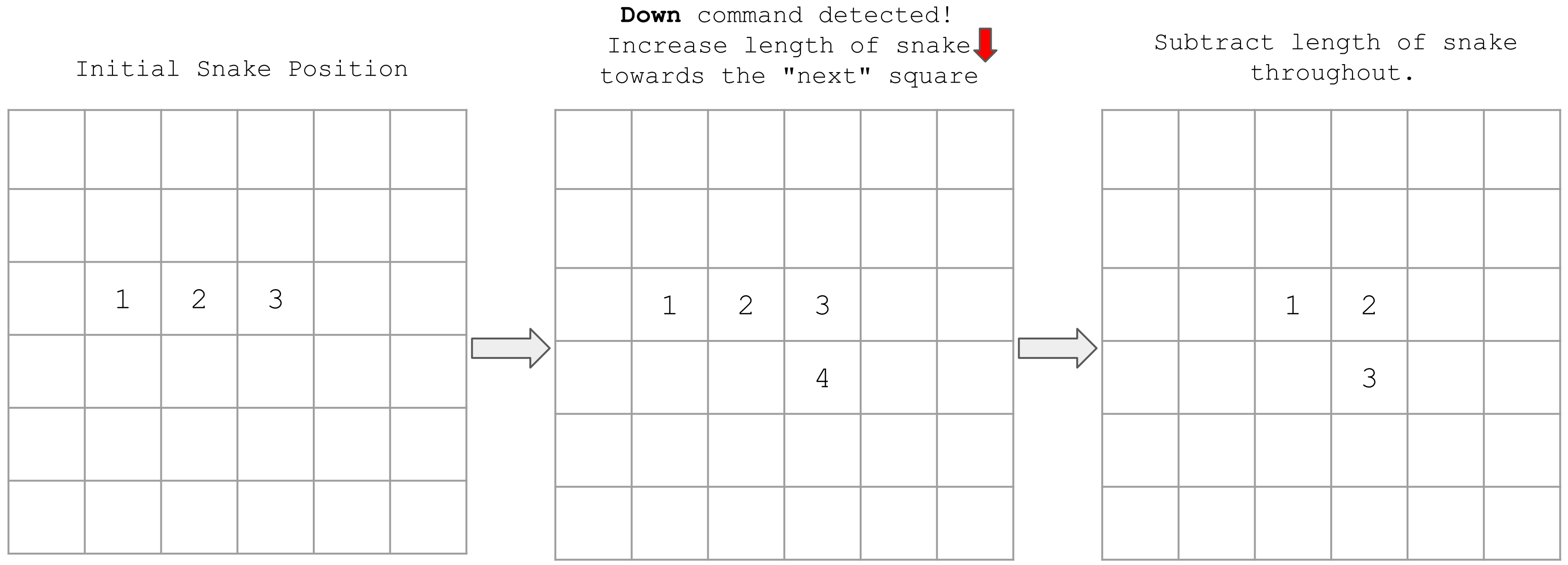
```
# # # # # # # # # # # # # # # # # # # # # #
# 0 0 1 2 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
# # # # # # # # # # # # # # # # # # # # # #
```

```
# # # # # # # # # # # # # # # # # # # # # #
#     * * @                               #
#                                         #
#                                         #
#             $                           #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
#                                         #
# # # # # # # # # # # # # # # # # # # # # #
```

How to emulate motion?

Let the largest number represent the length of the snake. Assign the head of the snake to be the length of the snake (and decrement it down its body)

Then, first extend the snake in the direction of motion, then reduce its length throughout



How to grow?

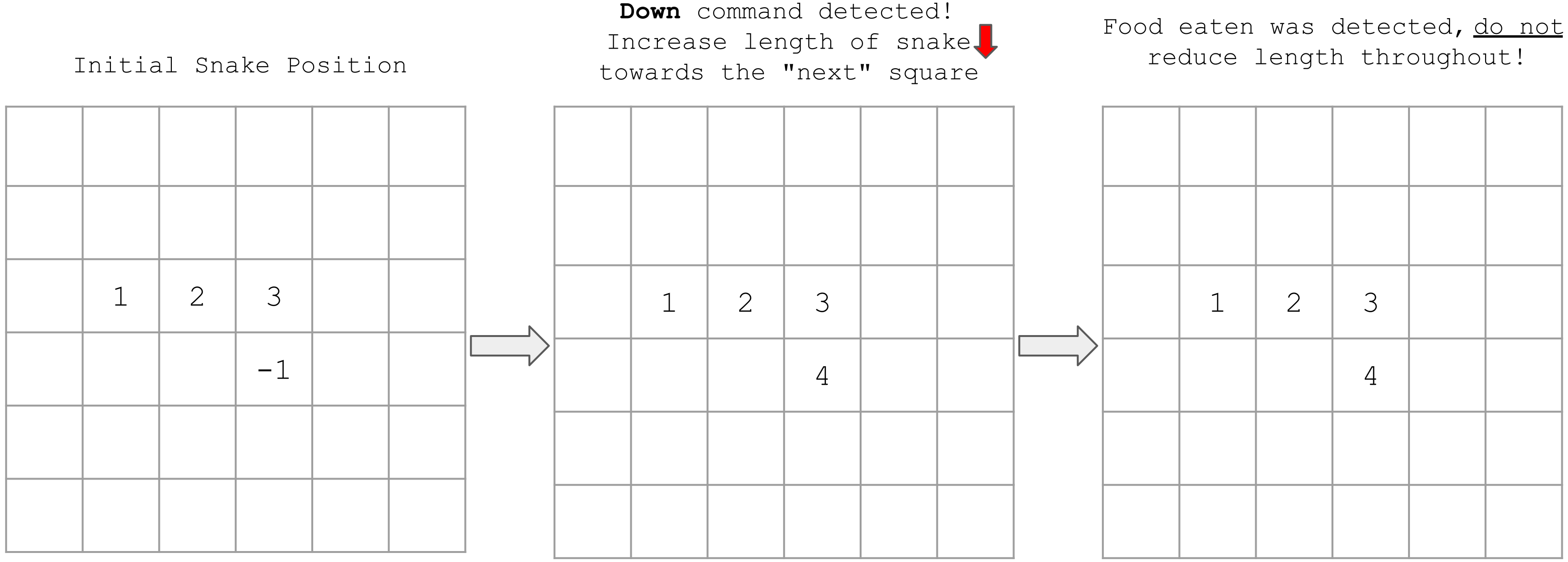
First extend the snake in the direction of motion.

If food was detected in that square, DO NOT reduce its length throughout

Initial Snake Position

Down command detected! Increase length of snake towards the "next" square

Food eaten was detected, <u>do not</u> reduce length throughout!

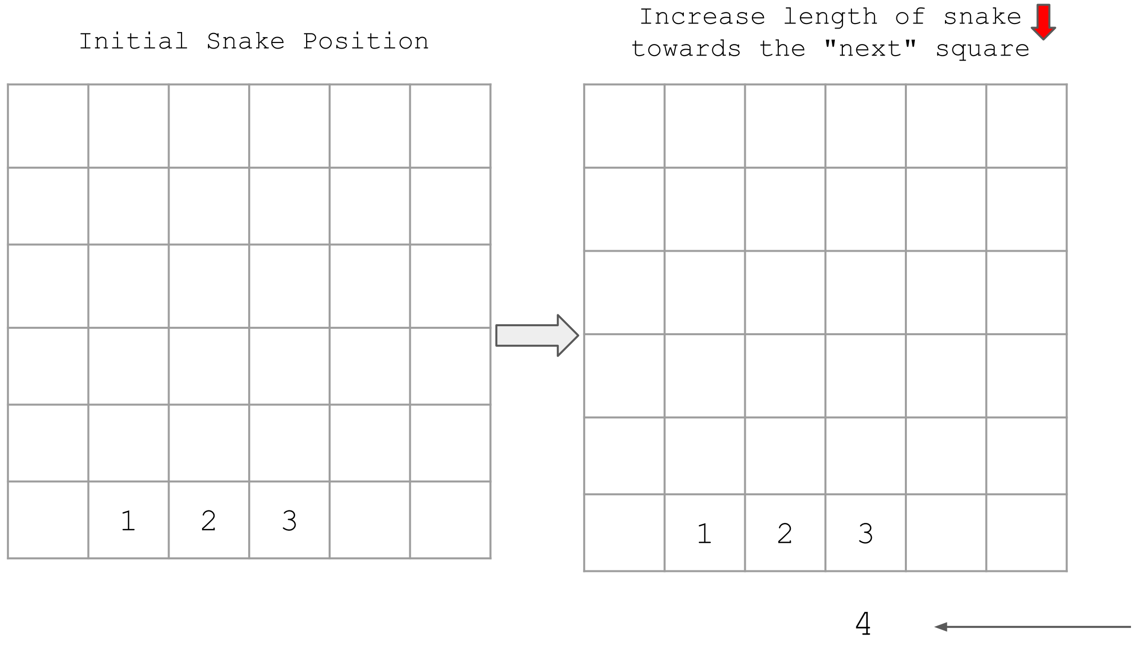


How to detect collision?

Check if projected head will collide with wall or self

Initial Snake Position

Down command detected!
Increase length of snake
towards the "next" square



Position of 'projected' head clearly exceeds map, therefore we terminate game!

How to generate food?

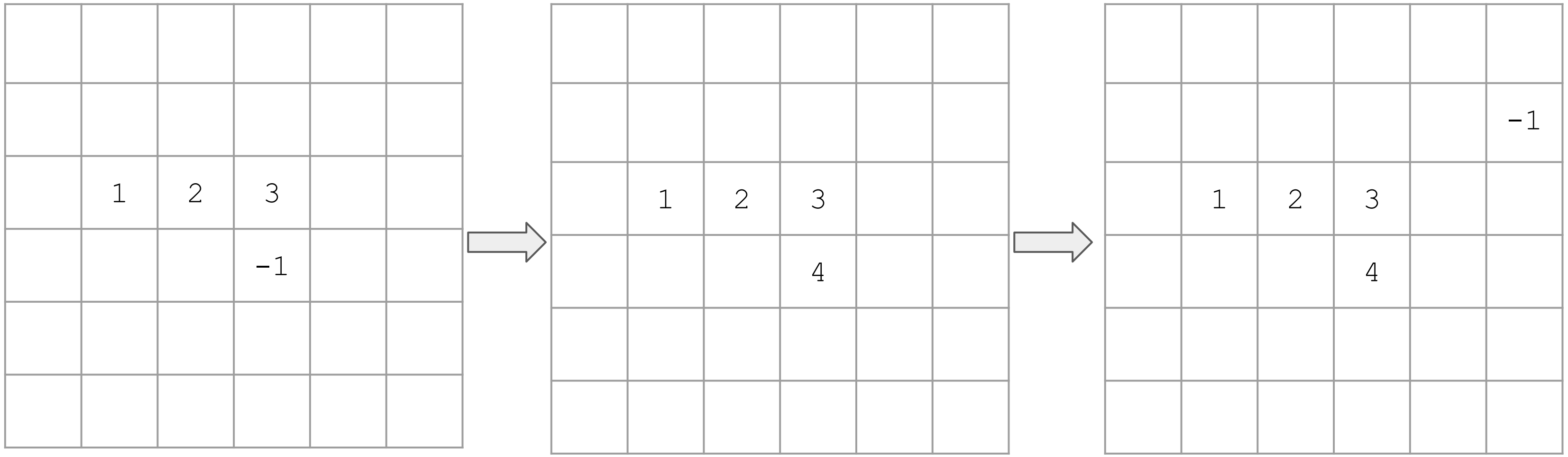
First extend the snake in the direction of motion.

If we detected food and head has collided, we proceed to generate food at any other random square
(Hint: using the rand() command)

Initial Snake Position

Down command detected! ⬇
Collision of head and food detected!

Randomly generate a food at
<u>any</u> empty square



Code Overview

```
int main() {

    srand(time(0));             // Initialize a random seed

    int map[H][W] = {};         // Playable map. 0: Empty square. +ve: Snake position, -ve: Food position.
    int direction = 3;          // Command direction, obtained from user input. 0: UP, 1: DOWN, 2: LEFT, 3: RIGHT
    double prevTime = clock();  // Record of clock from previous cycle to ensure

    int length = 3;             // Current length of snake
    int posHead[2] = { 0, 4 };  // Current position of snake's head
    int posFood[2] = { 0, 0 };  // Current position of food

    // Initialize snake and food position on map
    initializeSnake(map, posHead, length);
    generateFood(map, posFood); // TODO: Fill in this function!
```

Initialize a random seed based on current time

Outerscope variables that will be constantly changed in the subsequent while loop, including map, commanded direction, and positions of head / food

Initialize Snake's head and body, as well as generate a random food position on the map

Code Overview

```
while (1) {

    // Refresh console display
    clearScreen();
    render(map, posHead);
    std::cout << "Direction received = " << direction << "\n";

    // Hold program in this loop until timestep of loop passed or key pressed
    while (!isTimeElapsed(prevTime, TIMESTEP) && !isKeyPressed());

    // Get commanded direction
    if (isKeyPressed()) direction = getKeyDirection();

    // Update position of snake's head with commanded direction
    updateHeadPosition(posHead, direction); // TODO: Fill in this function!

    // TODO9: Check for collisions with walls and body.
    // If collision has occured, print "Game Over" and exit while loop.  (~3 lines of code)
    if (true)
    {
    }

    // Update map
    update(map, posHead, posFood, length);

    // Update length of snake and previous cycle timestamp
    length = map[posHead[0]][posHead[1]];
    prevTime = clock();
}

return 1;
```

Loop through this segment of code until game over.
If game over, break out of loop.

Clear the previous display & re-print the updated map

Wait here until either (a) Timestep has passed, or (b) key is pressed.
If either conditions is true, while condition will equal false and the program will move on to the next line. Note: ! is the NOT operator

If key was pressed in previous line, update the direction.
Otherwise, keep the previous known direction

From the commanded direction, 'project' where the new snake head will be

Check if projected head will lead to "Game Over" conditions

From position of projected head, check if snake should move/grow, should food be re-generated, and update the map accordingly

Update the length, which is the value of the head's position.
Also, update the time of prevTime to reset the initial counter for the isTimeElapsed() function.

Further Tips

INSTRUCTIONS

Complete all **TODO** functions in Snake.cpp

Recommended order of solutions:

1) render()
2) updateHeadPos(), extendSnake(), reduceSnake()
3) generateFood(), isCollisionFood()
4) update()
5) isCollisionSelf(), isCollisionWall(), & Game Over conditions in main()

Once completed, you may explore the following:

- Add a 4x4 obstacle at the middle of the map. Food should not be generated there.
- Add a wrap-around function for the walls (i.e. enter right wall, emerge left wall)
- Add a 'poison' that reduces length by half (while respecting minimum = 1)