


<b>Assignment Case</b>	
DS using CH5	
<b>Periode Berlaku</b> Semester Genap 2023/2024 <b>Valid on</b> Even Year 2023/2024	<b>Software Laboratory Center</b> <b>Assistant Recruitment 24-2</b>

**Note: Please focus on the main logic and main feature!**  
*(Splash screen and design are not scored)*

## Soal

Case

### Drift Snake

You've been hired to develop Drift Snake, where entertainment meets education! In this project, you'll be implementing advanced data structures: using a queue for snake growth and a circular double-linked list for dynamically changing the game map. Get ready to blend fun with learning as you code your way through this exciting gaming adventure!

#### ➤ Welcome Screen

- **This** menu contains 2 menus, which are **Play** and **Exit**.
- **Prompt** user to input chosen menu. **Validate** the input must **between 1 and 2 inclusively**.



Figure 1. Welcome Screen

1. If user choose **"Play"**, then:
  - **Change** the screen to the **Game Screen**.
2. If user choose **"Exit"**, then:
  - **Change** the screen to **Slash Screen**, then **terminate the program**.

## ➤ Game Screen

- This screen **shows the score, current map, and input field.**

```
Score : 1

#####
#           ###          #
#   #####   ###   #####   #
#   #                #   #
#   #   #####      #   #
#   #   #          #   #   #
#   #   #          #   #   #
#           .O
#   #   #          #   #   #
#   #   #          #   #   #
#   #   #####      #   #
#   #   *          #   #
#   #####   ###   #####   #
#           ###          #
#####

Input :
```

Figure 2. Game Screen

## ➤ Maps

```
#####
#           ###          #
#   #####   ###   #####   #
#   #                #   #
#   #   #####      #   #
#   #   #          #   #   #
#   #   #          #   #   #
#           #
#   #   #          #   #   #
#   #   #          #   #   #
#   #   #####      #   #
#   #   #          #   #
#   #####   ###   #####   #
#           ###          #
#####
```

Figure 3. Town Map

```

#####
#      #   ###   #      #
#      #   ###   #      #
#      #         #      #
#      #   #####  #      #
#      #         #      #
#      #         #      #

#      #         #      #
#      #         #      #
#      #   #####  #      #
#      #         #      #
#      #   ###    #      #
#      #   ###    #      #
#####

```

Figure 4. Plaza Map

```

#####
#              #####   #
#  #      #              #  #
#  #      #              #  #
#  #      #              #  #
#  #      #              #  #
#  #      #              #  #

#  #      #              #  #
#  #      #              #  #
#  #      #              #  #
#  #      #              #  #
#  #      #              #  #
#              #####   #
#####

```

Figure 5. Garden Map

- There are a total of **3 maps**: **Town, Plaza, Garden**.
- The **maps** are provided in **TXT files**. Just **copy** and **paste** the maps into your code.
- The **width** of each map is **30**, and the **height** of each map is **15**.
- When the game is **loaded**, **insert the maps** into a **circular doubly linked list**. Make sure to **insert the maps in order (Town – Plaza – Garden)**. This means that the **first spawn** map will be **Town**.
- **Create a current map** pointer variable and **assign** it to **point** to the **head** of the **circular doubly linked list**. This will be used to **display** the map on the **Game Screen**.

- Each map is surrounded by walls (#), but there will be an **empty space** in the **center left** and **center right** of the map.
- **Generate a food (\*)**, on a **random position**. **Validate** that the position is a **valid position** by checking if the position is an **empty space** and **not on the center left or center right**.
- There will only be **one food** at a time. The **current food** will be **replaced** every time it's **eaten** by the snake and every time the **current map** is **changed**.
- When the snake goes to the **center left empty space**, then :
  - **Move** the **current map** pointer to the **previous**.
  - **Clean** the **snake's body** and **food** from the **current map**. To do this, **reset** the **current map** to its **original** state.
  - **Move** the **snake's head** to the **right empty space** and **hide** the **snake's body** from the previous map by **changing the coordinates** of each snake's body to **(-1, -1)**.
  - **Generate a new food (\*)**, on a **random position**. **Validate** that the position is a **valid position**.
- When the snake goes to the **center right empty space**, then :
  - **Move** the **current map** pointer to the **next**.
  - **Clean** the **snake's body** and **food** from the **current map**. To do this, **reset** the **current map** to its **original** state.
  - **Move** the **snake's head** to the **left empty space** and **hide** the **snake's body** from the previous map by **changing the coordinates** of each snake's body to **(-1, -1)**.
  - **Generate a new food (\*)**, on a **random position**. **Validate** that the position is a **valid position**.

## ➤ Gameplay

- The snake's movement will be based on the concept of a **queue**. Make sure to use **double linked list** to represent the **queue**.
  1. Node Structure:
    - **Each node** in the **queue** will have its own **x** and **y coordinates** to represent the **snake's position** on the **current map**.
  2. Snake:

- When the game is **loaded**, **initialize** the **snake** by **inserting two nodes** into the **queue**.
  - The **first node** inserted will represent the **head** of the **snake**, and the **rest of the node** will represent the **snake's body**.
  - **Place** the **snake** at the **center** of the map with its **head** facing **right**.
3. Movement:
- **Declare** variables to store **horizontal** speed and **vertical** speed. These variables will be used to **control** the **snake's movement**.
  - When the user inputs 'w':
    - If the snake **is not** currently facing **downwards**, **change** its direction **upwards** by setting **vertical** speed to -1 and **horizontal** speed to 0.
  - When the user inputs 'a':
    - If the snake **is not** currently facing **right**, **change** its direction to the **left** by setting **horizontal** speed to -1 and **vertical** speed to 0.
  - When the user inputs 's':
    - If the snake **is not** currently facing **upwards**, **change** its direction **downwards** by setting **vertical** speed to 1 and **horizontal** speed to 0.
  - When the user inputs 'd':
    - If the snake **is not** currently facing **left**, **change** its direction to the **right** by setting **horizontal** speed to 1 and **vertical** speed to 0.
  - Every move:
    - **Update** the position of the **snake's head** by **pushing a new node** to the **front** of the **queue**, according to the current **horizontal** speed and **vertical** speed.
    - If the **snake moves** into an **empty space**, **pop** the **end** of the **queue** to maintain a constant snake length.
    - If the **snake moves** into a **food (\*)**, **update** the **score**, then **generate a new food** on a **random position**. **Validate** that the position is a **valid position**. The **snake length** will **increase** because the **tail** is not **popped**.
    - If the **snake moves** into a **wall (#)** or into its own body (.), **change** the screen to the **Game Over Screen**.

- Set the **initial** value of the **score** to 1.
- **Increment** the **score** by 1 every time the **snake head** is on the **food**.

- This screen **shows the final score.**

**Figure 6. Game Over Screen**

"Inspired by passion, driven by purpose, together we shatter limits and redefine boundaries."

**Please run the EXE file to see the sample program.**