


Project Case	
DS Using C Project	
Periode Berlaku Semester Genap 2024/2025 Valid on Even Year 2024/2025	Software Laboratory Center Assistant Recruitment 25-2

Note: Please focus on the main logic and main feature!

(Splash screen and design are not scored)

Soal

Case

MR.TETRIS

In a world shattered by chaos, only MR.TETRIS can restore order. Once a harmonious land, Tetronia fell into disarray when the malevolent Lord Discordus unleashed the Chaos Core, turning the world into a shifting labyrinth of falling blocks and destruction. Armed with the TetroGlove, MR.TETRIS must travel across five fallen districts, rebuilding fractured landscapes, stabilizing crumbling towers, and battling the minions of chaos. By strategically placing tetrominoes, he clears corruption, creates pathways, and unlocks hidden secrets. He inches closer to the final challenge as he gains new abilities like TetroShift and Line Burst. With time running out, only the sharpest minds can master the challenge of MR.TETRIS!

➤ Home Page

- Read player's data from **user.txt** then put it into a **hash table** of **size 27**.
- The **user.txt** file contains **username**, **password**, **score**, **cleared row**, and **total block** separated by **' , ' (comma)**.
- The **hash table** uses the first character of the **username** for the **key** without considering **uppercase** or **lowercase** letters. For example, **"Denny"** and **"deddyanto"** will have **three** as its key. If the username starts with a **symbol**, it will have **26** as its key.
- If collusion happens, use **chaining** that implements a **single-linked list** that sorts the user based on the **player's score** in **descending** order.
- This menu contains 3 menus: Play, **View Player**, and **Exit**.
- **Prompt** user to **input chosen menu**. **Validate** the input must be **between 1 and 3 inclusively**.



Figure 1. Home Page

1. If the user chooses **Play (Menu 1)**, then:
 - **Prompt** the user to **input a username**.

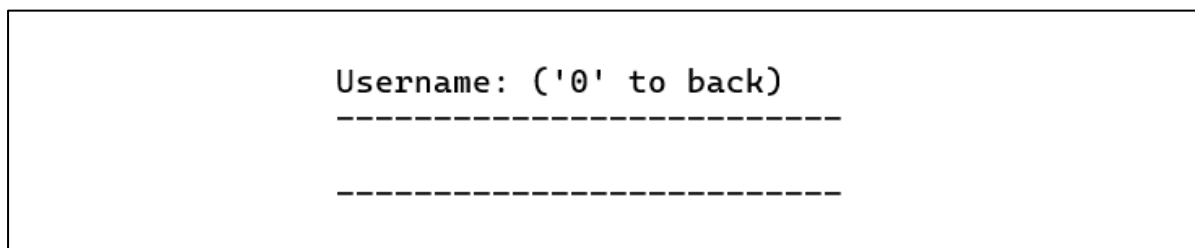


Figure 2. Input username

- If the user types "**Admin**", redirect to the **admin page**.
- If the user types "**0**", redirect **back** to the **home page**.
- **Validate** username **cannot be empty**.
- **Validate** username's length must be between **4-20 (inclusive)**.
- **Validate** username can only contain '**.**' (**full stop**), '**-**' (**dash**), and '**_**' (**underscore**).
- If the username **does not match** the criteria, **show an error message** to the user based on the error.

```
Username: ('0' to back)
-----

-----

*username must between 4-20 (inclusive)
```

Figure 3. Example of the error message

- If the username **does not exist** in the **hash table**, ask the user to **register**. Otherwise, ask the user to **log in**.

```
User already registered! Want to login [Y]?
```

Figure 4. Display for registered user

```
Register new user! Want to register [Y]?
```

Figure 5. Display for unregistered user

- If the user **does not** press '**y**' (in case sensitive), redirect to **Home Page**.
- If the user presses '**y**' (in case sensitive), prompt the **user** to input a **password** (must be displayed as '*').
- If the user is already **registered** in the system, **validate** that the password is the same as the **password** in the **hash table**. Otherwise, show error messages.

```
Password: ('0' to back)
-----

-----

*Incorrect password
```

Figure 6. Display password input for registered user

- If the user is not **registered** in the system, **validate that** the password meets these criteria.
 - Has **length** between **8-24 (inclusive)**.
 - Contains **uppercase** characters.

- Contains **lowercase** characters.
- Contains **symbols** or **numeric** characters.

If a **condition** is met, give 'v' to mark that the condition.

Password: ('0' to back)

[v] Length 8-24 (inclusive)

[] contains upper case character

[v] contains lower case character

[v] contains symbol or numeric character

Figure 7. Display password input for unregistered user

- If the user choose **View Player (Menu 2)**, then:
 - **Show** all data from the **hash table**. Implement **pagination** such that **one page** only contains **17 users**.
 - If the user presses '**d**', navigate to **next page**. **Validate** that the user is **not** on the **last page**.
 - If the user presses '**a**', navigate to **previous page**. **Validate** that the user is **not** on the **first page**.
 - If the user presses '**q**', redirect back to the **Home Page**.

No.	Username	Score	Clear	Block
035	michael	990	9	55
036	mason	640	7	40
037	mia	230	2	16
038	noah	710	7	39
039	olivia	0	0	13
040	plmko	580	5	41
041	sophia	580	6	30
042	sarah	200	2	17
043	victor	320	3	20
044	william	860	8	48
045	zoe	410	4	22
046				
047				
048				
049				
050				
051				

Input 'a' and 'd' to navigate ('q' to exit):

Figure 8. View Player Page

3. If the user chooses **Exit (Menu 3)**, then:
 - Ask for **confirmation** from the user, if the user presses '**y**' (**case insensitive**), then **terminate** the program. Otherwise, redirect to the **Home Page**.

Are you sure [input 'Y' to confirm (case insensitive)]?

Figure 9. Confirmation message

- Write user data to **user.txt**.

➤ Game Page

- Before the game starts, read all blocks from **block.txt**. The blocks are written with '**#**' (**hash tag**) and '**.**' (**full stop**) representing **empty space**. Each block is separated by '**=**' (**equal sign**).

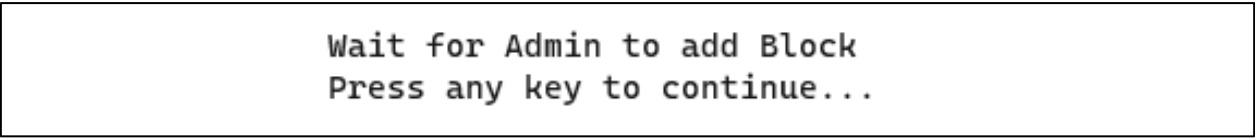


Figure 12. Error message for the user because no block yet

1. **Game Menu**

- Display the **game page**, with all these components.

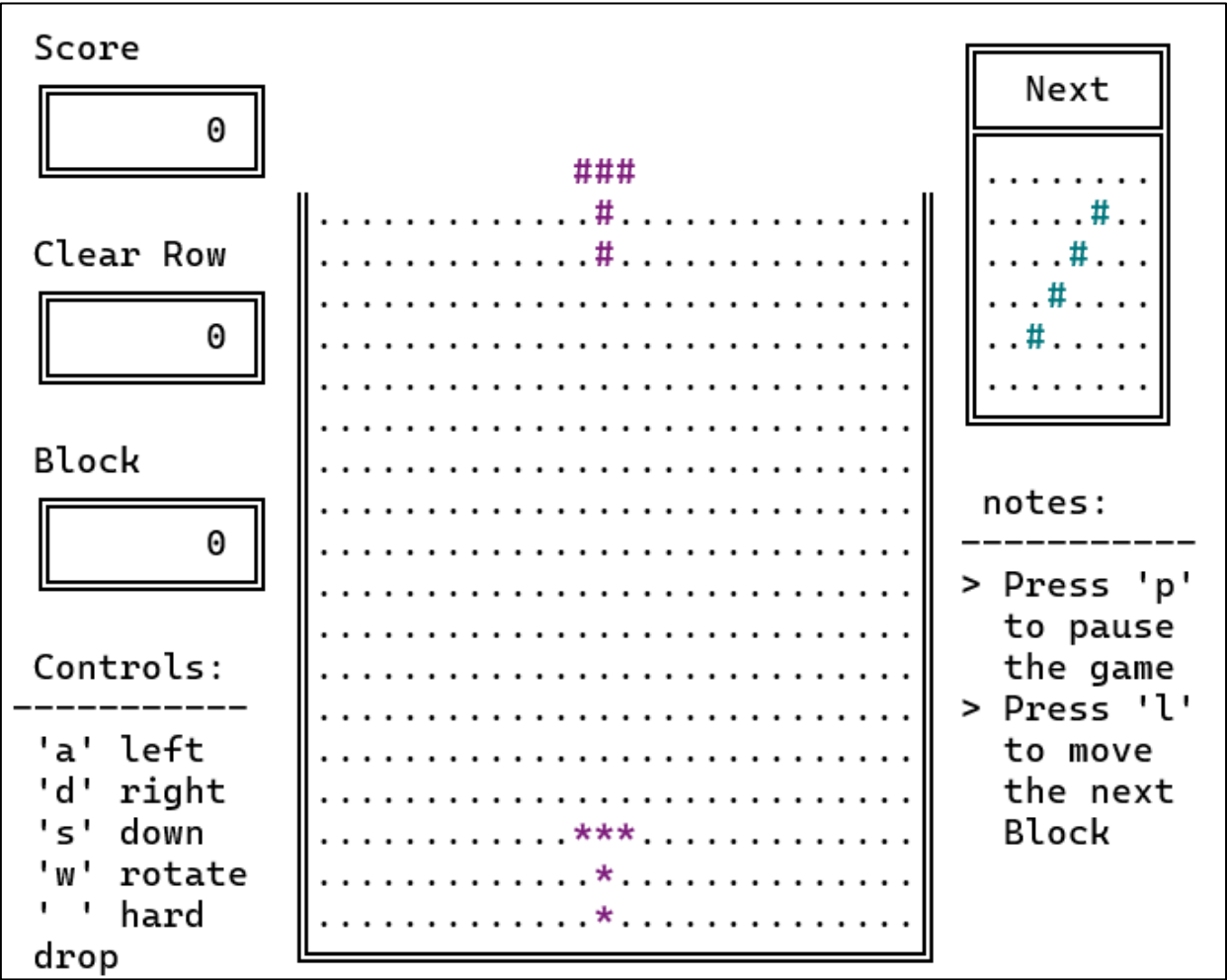


Figure 13. Display of the game menu

- **Box container**, which has **28 x 19** as its dimensions.
- **Current block** that represented with '#' and its **preview** that represented as '*'.
- **Statistic preview**, that shows the current player's **score**, **cleared row**, and **placed block**.

- **The next block preview**, consists of **five** blocks that **are chosen randomly** from the **block's double-linked list**. The next blocks are stored in a **single circular-linked list**.
 - **Notes**, to tell the user to press '**p**' to **pause** the game and '**l**' to **change the next block**.
 - **Controls**, to tell the user to use '**w**', '**a**', '**s**', '**d**' and '**'**' to control the current block.
- Spawn a **new block** to the **box container** such that the **bottom of the block** is the **top** of the **first container**.
 - Spawn current **block preview** at where the **current block** lands.

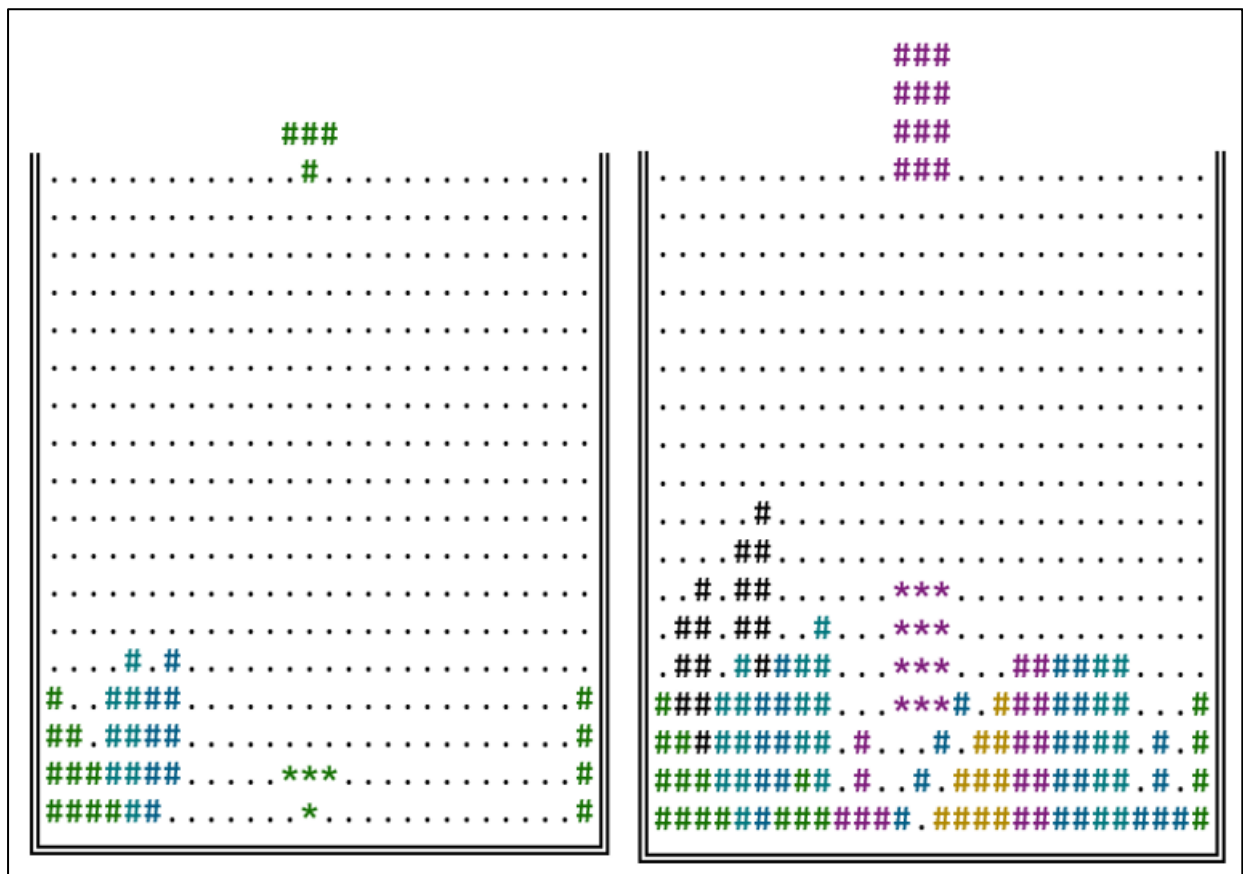


Figure 14. Illustration block spawn system

- Every $-\sqrt{15 \times (\text{number of placed block})} + 1000$, move the **current block** down by **one row**.
- If the user presses '**a**', then move the **current block** to the **right** by **one column**. **Validate** such that the block **cannot pass** through the **wall**.

- If the user presses 'd', then move the **current block** to the **left** by **one column**. **Validate** such that the block **cannot pass** through the **wall**.
- If the user presses 's', then move the **current block** down by **one row**. **Validate** such that the block **cannot pass** through the **wall**.
- If the user presses ' ' (**space**), then move the **current block** down until it reaches the **bottom** of the **box container** or **touches** another **placed block**.

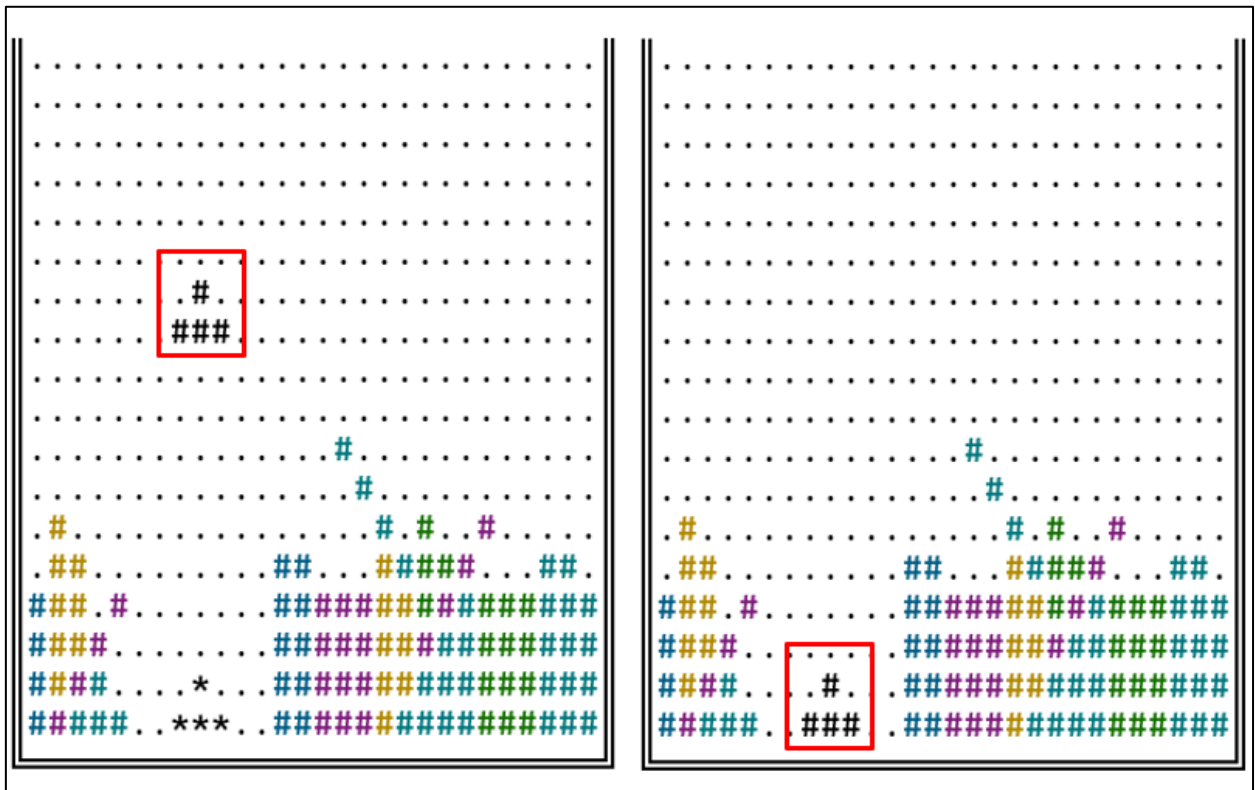


Figure 15. Illustration about how the block spawns

- If the user press 'l', change the **next placed block** to its **next block** in the **linked list**.

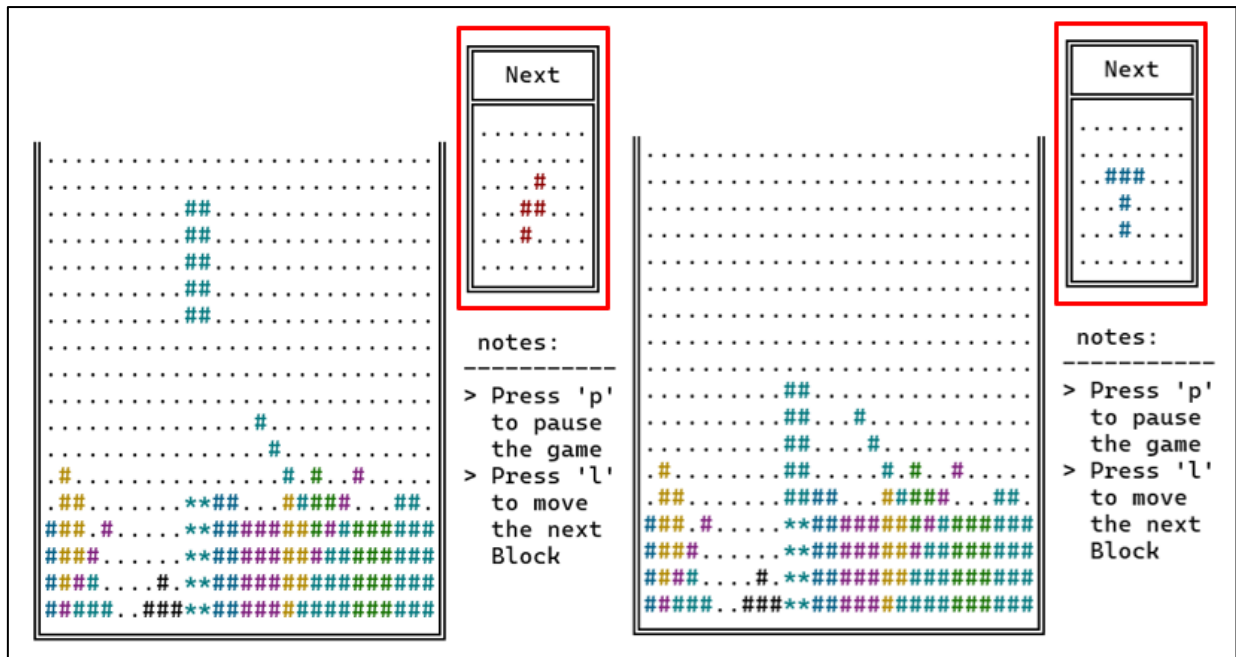


Figure 16. Illustration changes next placed block system

- If the user presses '**p**', **pause** the game and **ask** the user whether they **want to exit** or **not**. If the user presses '**y**' (in case sensitive), show the **game over menu**. Otherwise, **continue** the game.

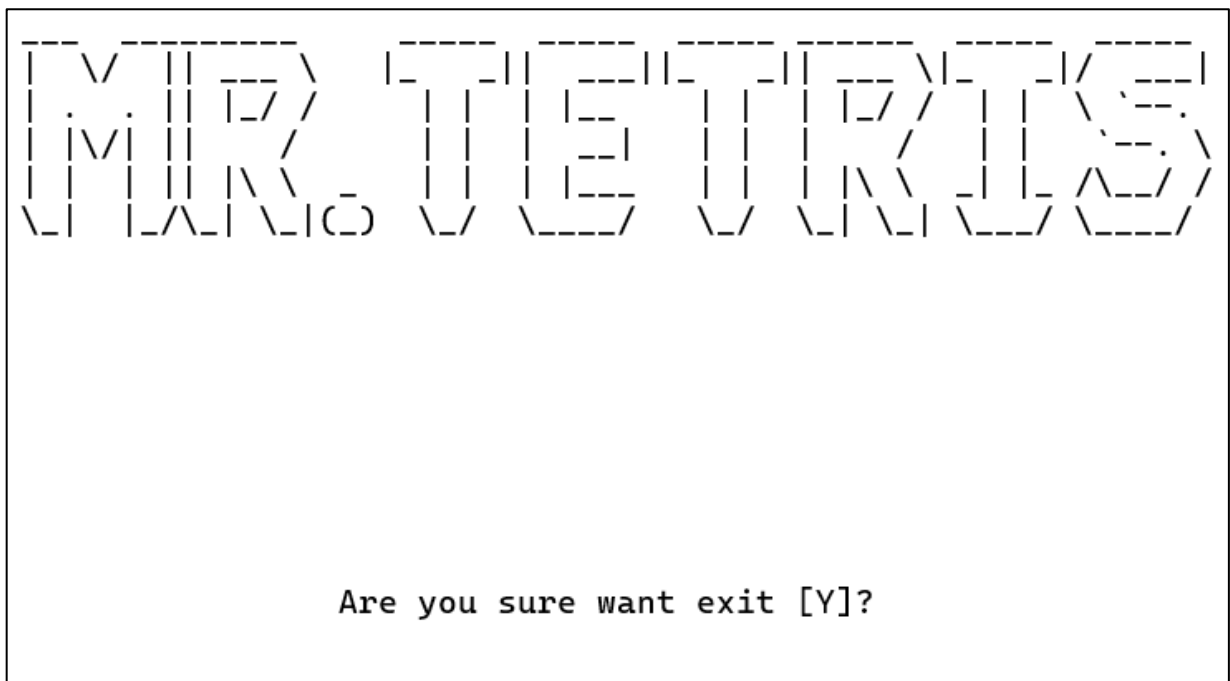


Figure 17. Display pause menu

- If the user presses '**w**', **rotate** the block from its **center point** by **90 degrees (clockwise)**.
Validate the rotation is only possible if it won't **collide** onto any **wall** or **placed blocks**.

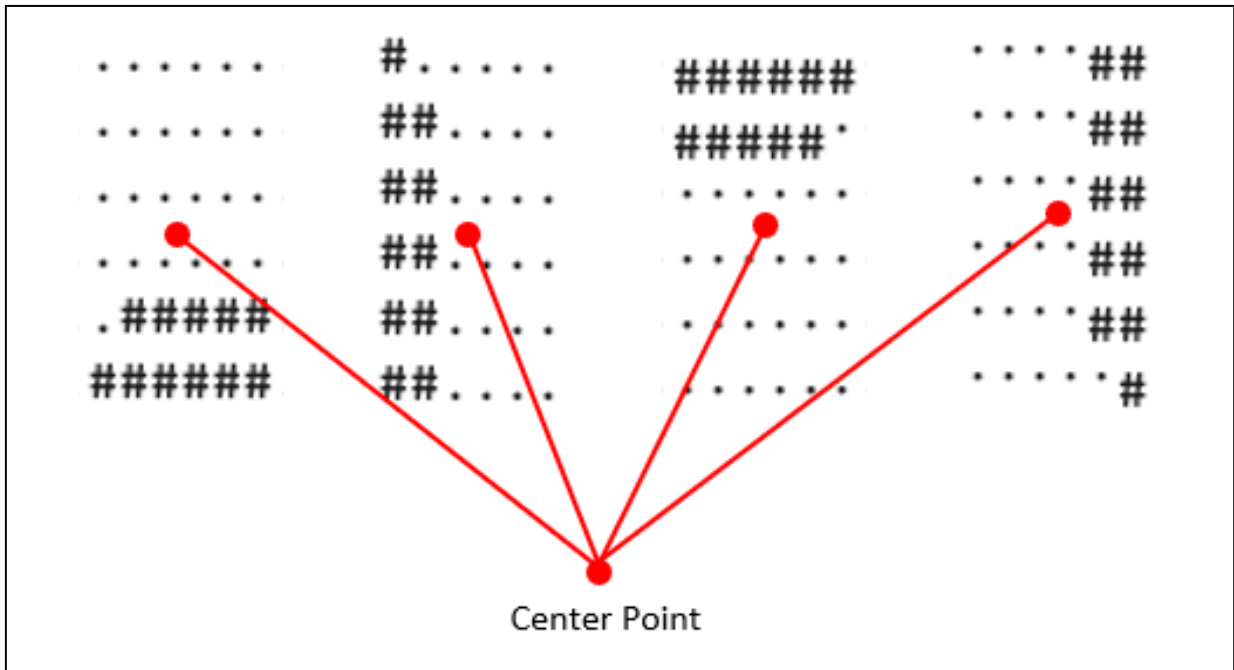


Figure 18. Illustration of how the block rotates

- If the **current block** cannot **move down** anymore (due to **reaching the bottom** of the **box container** or **touching** another **block**), spawn the **selected next block**, then **remove** the **selected next block** from the circular-linked list. Then **insert** a new block to the **circular-linked list** and **increase** the number of **placed blocks** by **one**.
- When **blocks spawn**, if the **first row** of the **box container** contains any '**#**', then show the **game over menu**.



Figure 19. Illustration of the losing condition

- If there are **one or more rows** that only contain '#', **clear** those rows and **move all above rows down** by the number of **the cleared rows**. Then increase the user's **cleared row** and the **user's score** by $\frac{n}{2} \times (200 + (n - 1) \times 20)$, where **n** is the **current cleared row**.

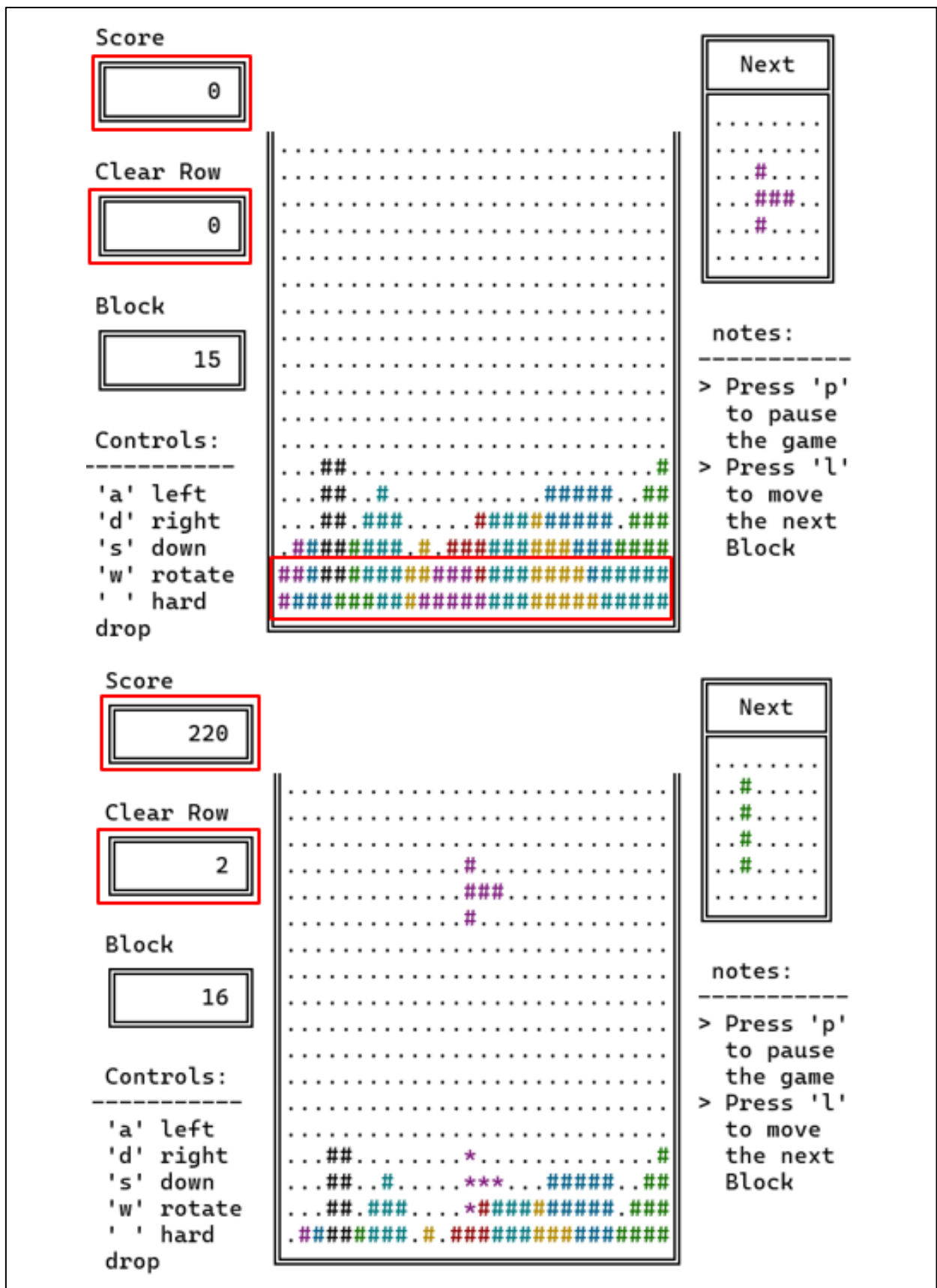


Figure 20. Illustration of cleared row conditions

2. Game Over Menu

- Display the **game over menu**. Prompt the user to press **enter** to **continue**.

```
You lost...
Press enter to continue...
```

Figure 21. Display of game over menu

- If the logged user is admin, **remove** all blocks from the **double-linked list**, then redirect back to the **Admin Page**.
- Display the user's **current** and **previous statistics**.

Current Stats	Previous Stats
Final score : 360	Final score : 820
Cleared row : 3	Cleared row : 7
Block count : 34	Block count : 60

!!Previous data will be lost!!
Do you want to save your stats [Y | N]?

Figure 22. Display of user's current and previous statistics

- If the user presses '**y**', update **user statistics** to **current statistics**. If the user presses '**n**', do not save the user's **current statistics**.
- Delete **All blocks** in the **circular-linked list** and the **double-linked list**. Then redirect to the **Home Page**.

➤ Admin Page

- Read all **block** from **block.txt**. Then insert it into the **double-linked list**.
- This menu contains 4 menus, which are **Play Game**, **Add new Block**, **Remove Block**, and **Log Out**.

- Prompt user to input chosen menu. Validate the input must be **between 1 and 4 inclusively**.

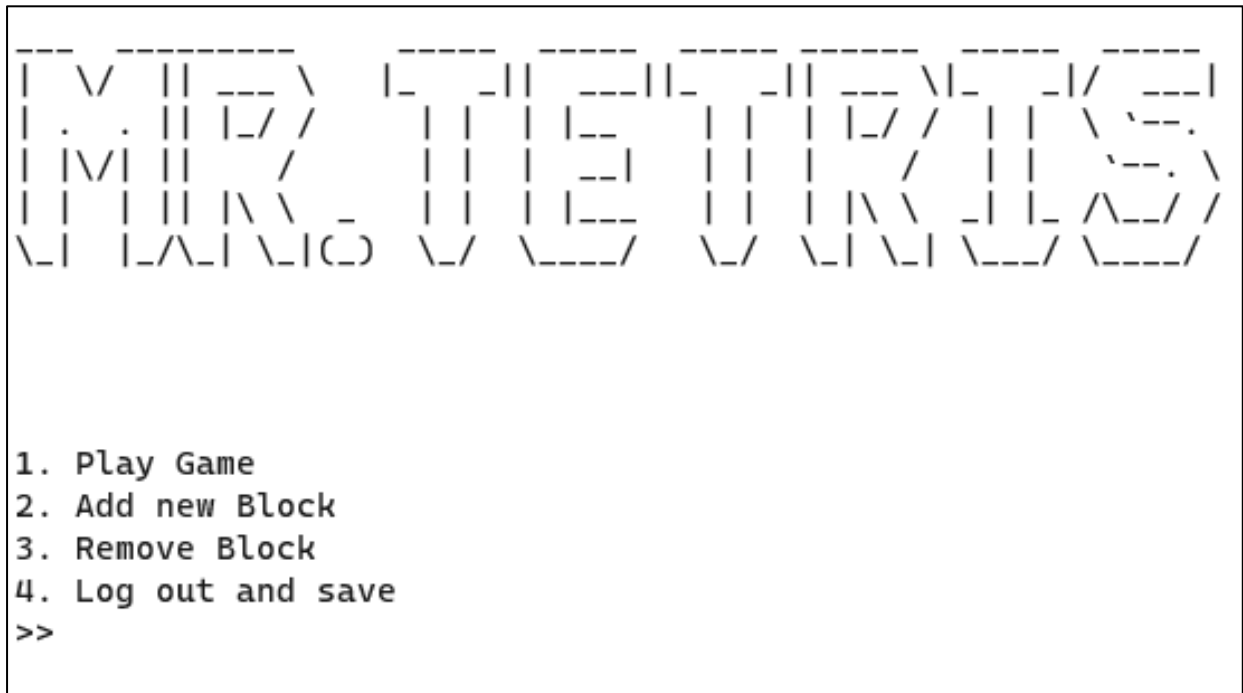


Figure 23. Display of user's current and previous statistics

1. If the admin chooses **Play Game Menu (Menu 1)**, then enter the **Game Page**.
2. If the admin chooses **Add New Block Menu (Menu 2)**, then:
 - Show **canvas** with a dimension of **6 x 6**, to let the admin **draw a new block**.

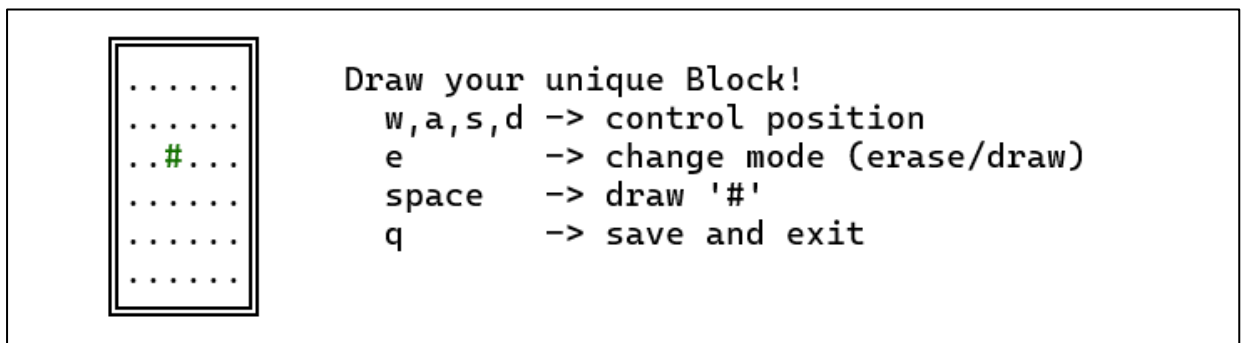


Figure 24. Display of new block canvas

- If the admin presses '**w**', move the brush **one block** to the **top**. Validate that the **brush** is still in **canvas**.
- If the admin presses '**a**', move the brush **one block** to the **left**. Validate that the **brush** is still in **canvas**.

- If the admin presses 's', move the brush **one block** to the **bottom**. Validate that the **brush** is still in **canvas**.
- If the admin presses 'd', move the brush **one block** to the **right**. Validate that the **brush** is still in **canvas**.
- If the admin presses 'e', change the **brush** to an **eraser** so the admin can delete '#'.

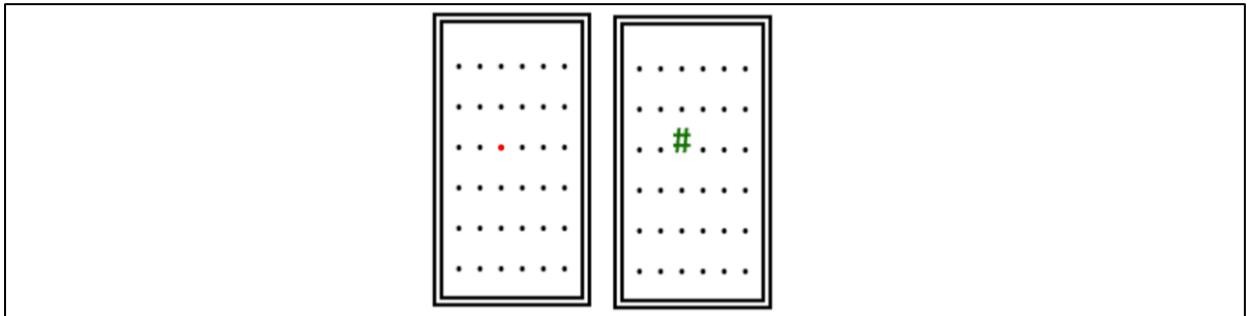


Figure 25. Illustration of changes of brush to eraser

- If the admin presses ' ' (space), then draw '#' or '.' (depending on current tools that the admin uses).

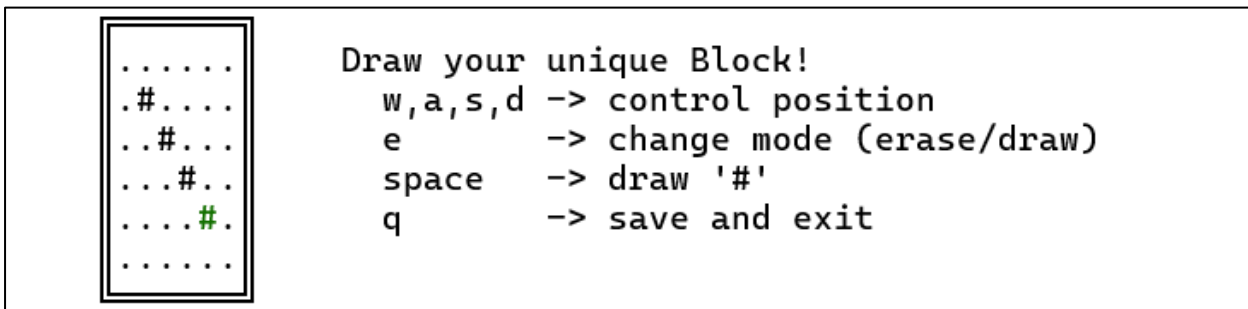


Figure 26. Illustration of drawing new block

- If the admin presses 'q', insert a **new block** to the **block's double-linked list**. Validate that the **new block** cannot **be empty**. Then redirect back to the **Admin Page**.

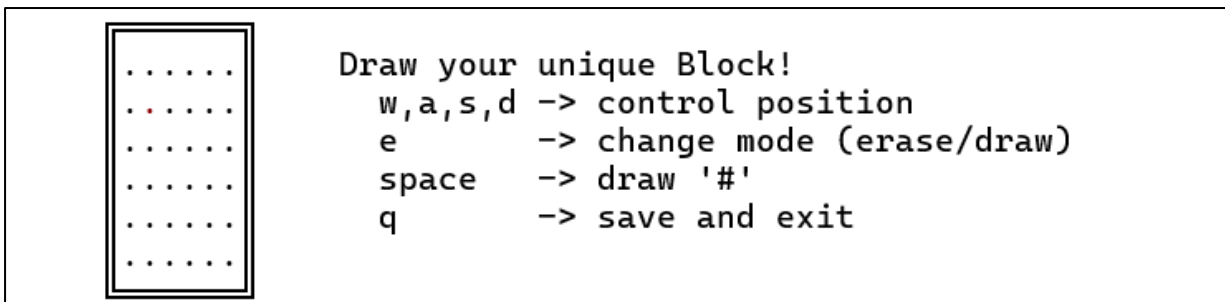


Figure 27. Error message when the new block is empty

3. If the admin chooses **Delete Block Menu (Menu 3)**, then:

- **Validate** that there are blocks in the **block's double-linked list**. If there is **no block**, display an **error message**.

```
No block yet...
Press enter to continue...
```

Figure 28. Error message when no block exists

- If any **block exists**, display the block from the double-linked list **sorted** from **smallest area to largest area**.

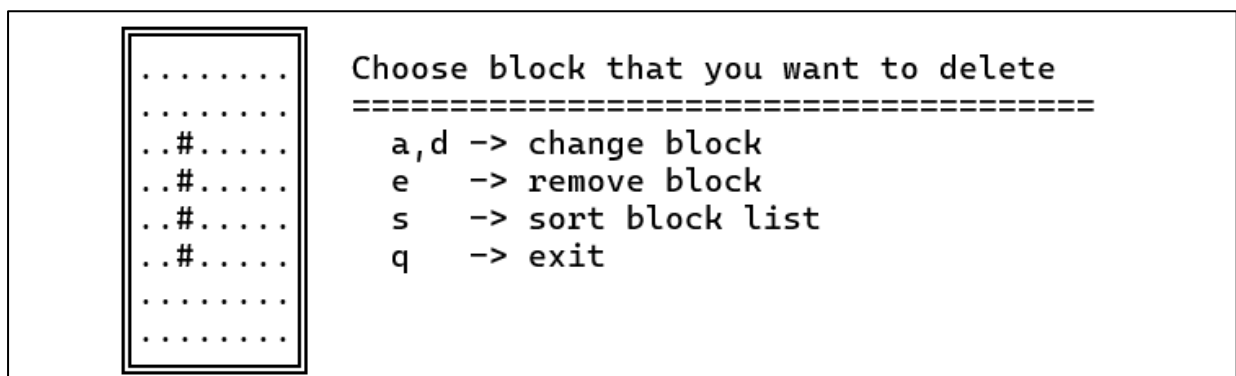


Figure 29. Display of Delete Block Menu

- If the admin presses '**a**', show **the previous** block in the **double-linked list**. Validate that the **previous block exist**.
- If the admin presses '**d**', show **the next** block in the **double-linked list**. Validate that the **next block exist**.
- If the admin presses '**s**', sort the **double-linked list**. If the block is sorted in **ascending order** sort it in **descending order** and **vice versa**.
- If the admin presses '**e**', show a **confirmation** message. If the admin presses '**y**', **delete** the selected block from the **double-linked list**.

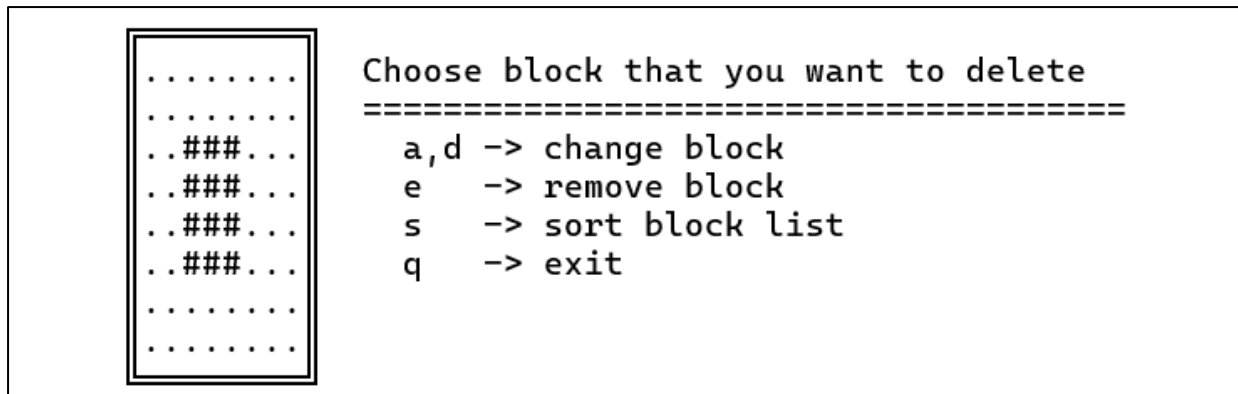


Figure 30. Display of confirmation message

- If the admin presses 'q', redirect back to **Admin Page**.
4. If the admin chooses **Delete Block Menu (Menu 3)**, then:
- Save all **blocks** from the **double-linked list** to **block.txt** with the same format as you read **all blocks**.
 - **Remove** all blocks from the **double-linked list**.

Please run the EXE file to see the sample program.

Komponen Penilaian

Scoring Component

No	Component	Weight
1	Input Output Menu	3%
2	Login Register	6%
3	File Manipulation	6%
4	Data Structure	35%
5	Gameplay	40%
6	Admin	10%