PP Final Project

3rd Week

Notice

Notice

- If **plagiarism** (copying all or some of another person's code) is detected, the incident will be reported to **Prof. Lee** and **disciplinary committee**(징계위원회).

Additional Notice – Last Week

- Project 1st Week submission due date has been extended.

  → Submission due date for
  both Project 1st Week & Project 2nd Week is now 12/4(Tue) 14:00.

- For Project 1st Week, no further delay allowed after 12/4(Tue) 14:00.
- For Project 2nd Week, delay deduction is 24h(-25%), 48h(-50%).
  (same as all prev. weeks)

Notice

- You also have practice lecture on Thursday.
- Only homework problems will be added.
- You can do both project and homework.

Notice

- Project 3rd Week submission due date
  →12/11(Tue) 14:00

- Delay Policy is same as usual – 24h(-25%), 48h(50%).

- We will provide you with sample answer for 1st Week.

Submission

- Submit by E-mail pp20182ta@gmail.com

- Email title format: [project] week# <student id> <name>

  example) [project] week3 2018-12345 홍길동

- Submit 1 file: `20xx-xxxxx.c` (Use your student ID as filename)
  - If your implementation is intentionally different from the given specifications (for a good reason), or if additional features are included etc., please specify it for us in the email.
  - If your implementations are same as given specifications, there's no need to write it.

Please follow the submission format

- Please follow the designated submission format (in previous slide).
- From now on, points will be deducted if you do not follow it.

- Ex) Points will be deducted if:
  - email title does not follow the designated format.
  - filename of code is not your student id.

- Also, please just attach as **.c** file. **No need to zip it.**

Notice

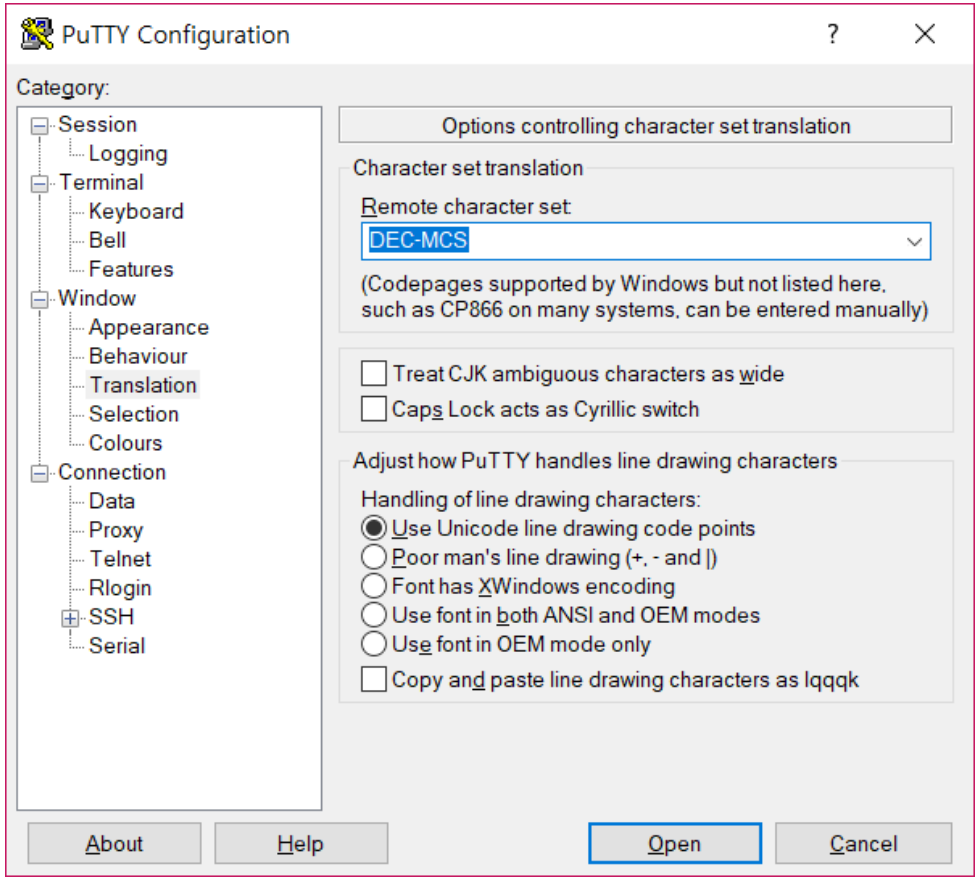
1. When drawing boxes please use **`addch`** or **`mvaddch`**, not **`printw`**.

2. Using ACS_VLINE and ACS_HLINE draws the boxes more smoothly than when using '|' and '–'. (They are different sets of characters.)

3. When exiting the program, **`endwin()`** *needs* to be called!!
   (If you **`return;`** prematurely in **`main()`** without calling **`endwin()`**, then console gets buggy. This makes it difficult for us to grade.)

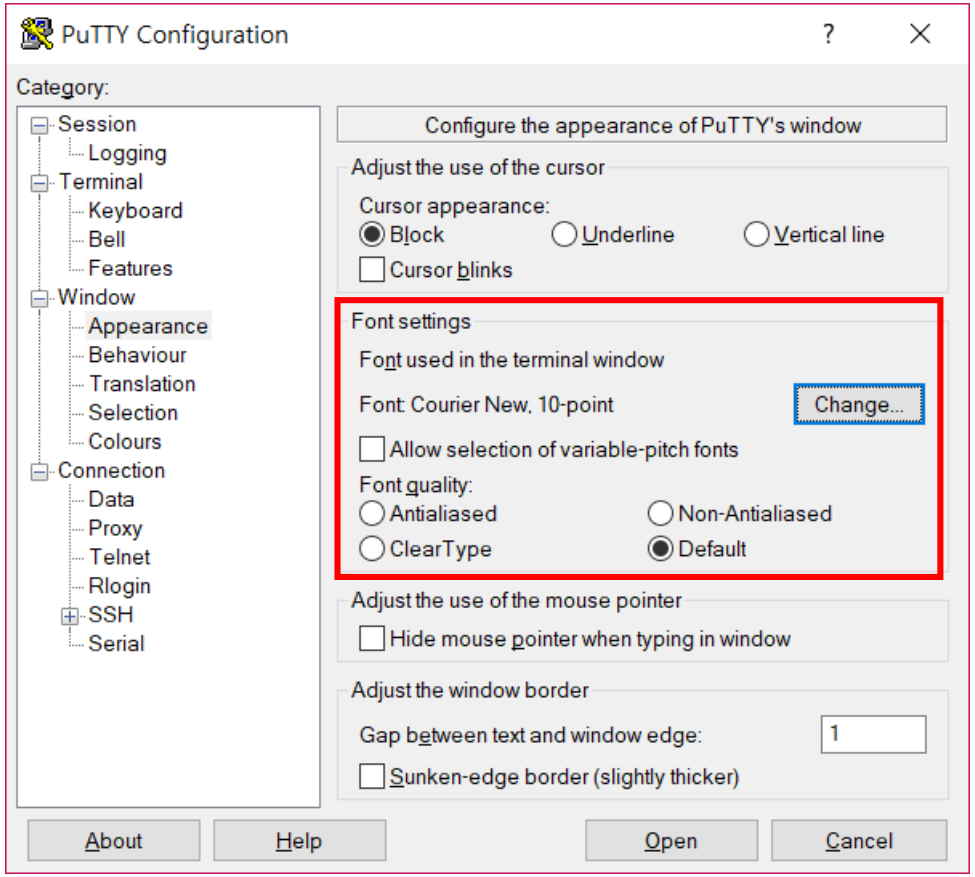
PuTTy Settings

- Window → Translation
- Set Remote character set as ‘DEC-MCS’



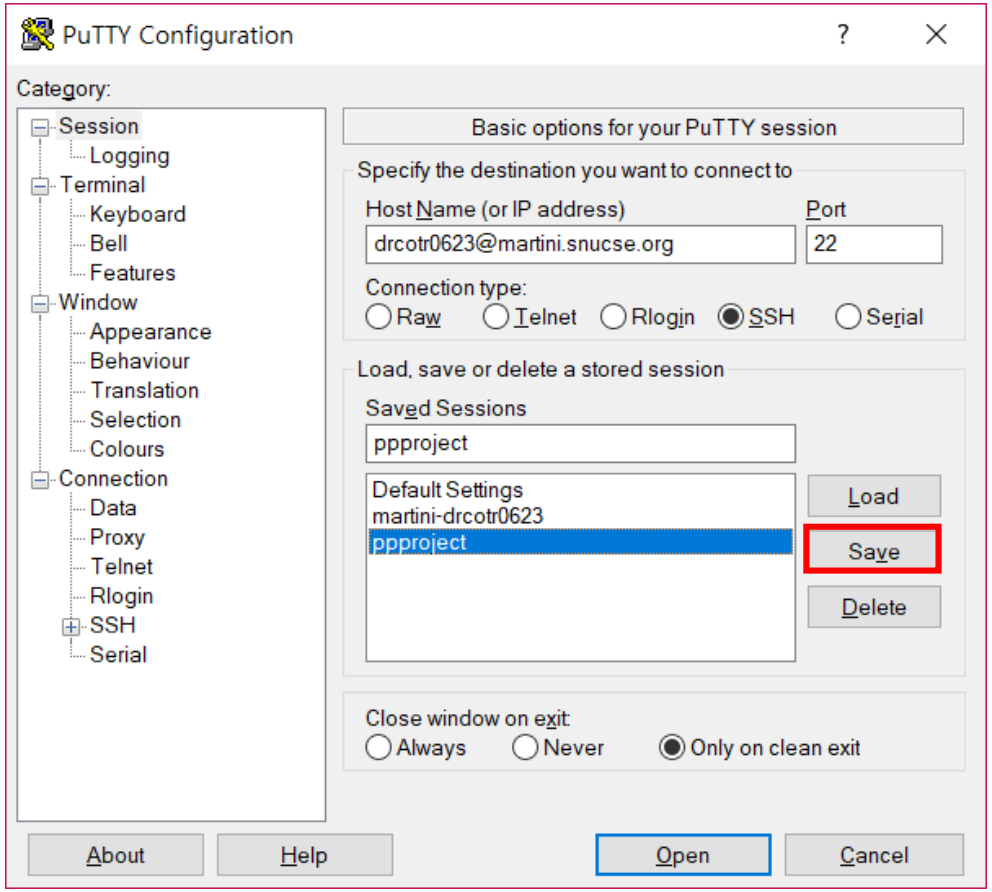
PuTTy Settings

- Window → Appearance
- You can change font.



PuTTy Settings

- Go back to Session
- You must save your setting to reuse.



DEMO

For this week

No skeleton

- No extra skeleton will be provided.
- Build on from your previous code.

- However, some tips for implementation will be provided!

Tasks for 3rd week

- Game Screen :
  1. You must complete implementing actual game.
  2. You must be able to choose two cards, check and remove them if they correspond to each other.
  3. You must implement pause option.
  4. You must display victory after you finish the game.

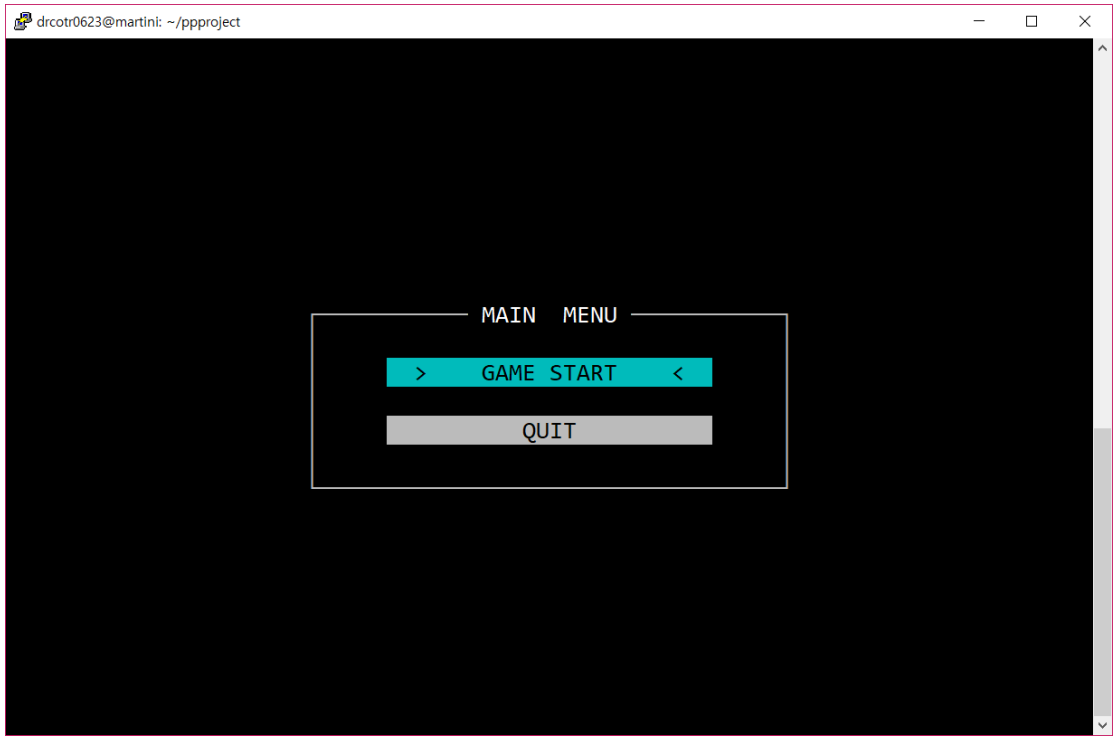
Tasks for 3rd week

- Additional implementation :
  1. You can add additional options to your game.

  2. Implement whatever you want. ex) Leaderboard, Option

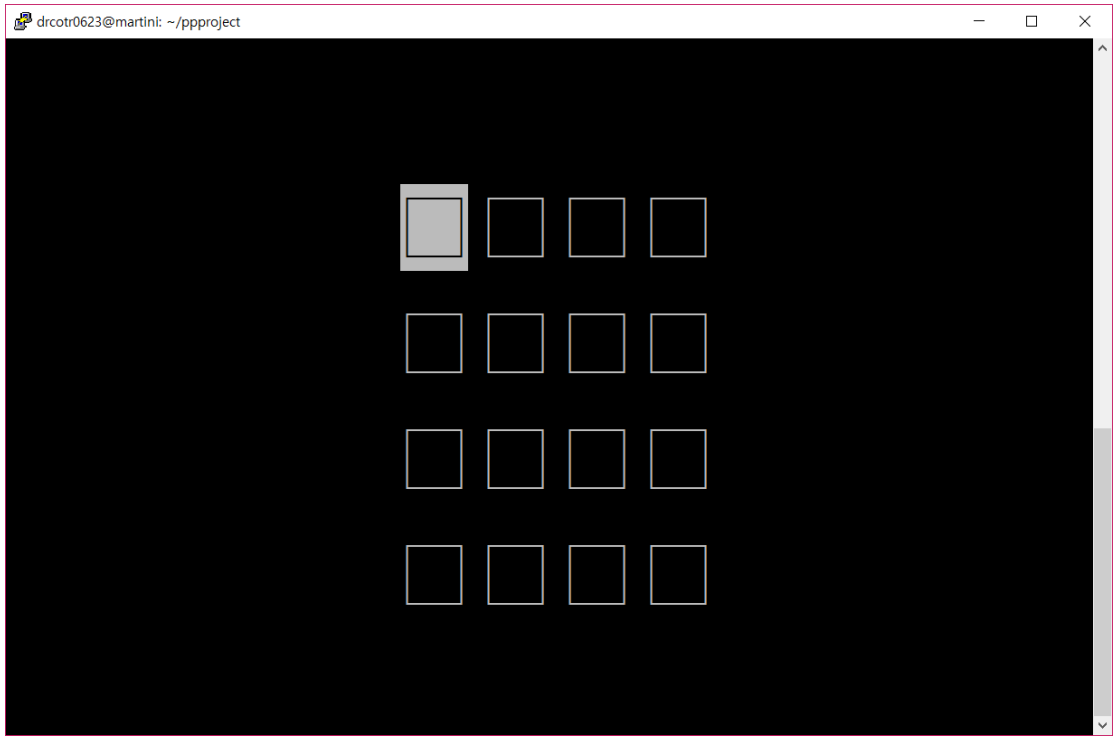
Examples

1. Main Menu



Examples

2. Game Screen

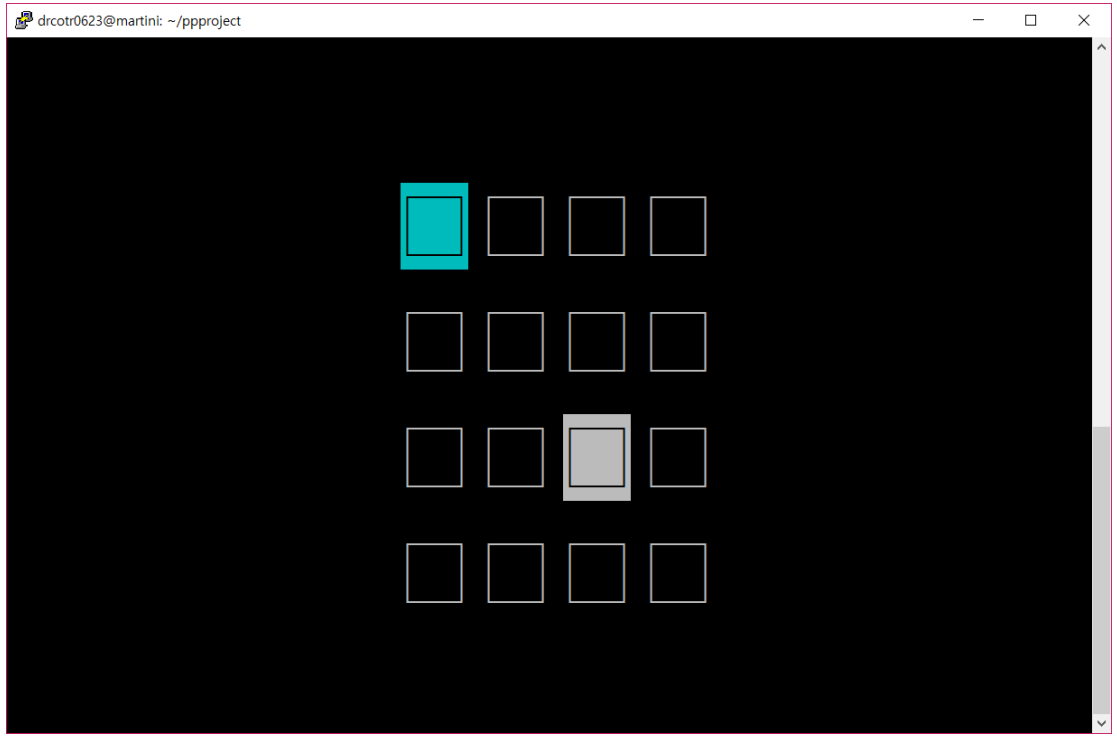


Examples

2. Game Screen

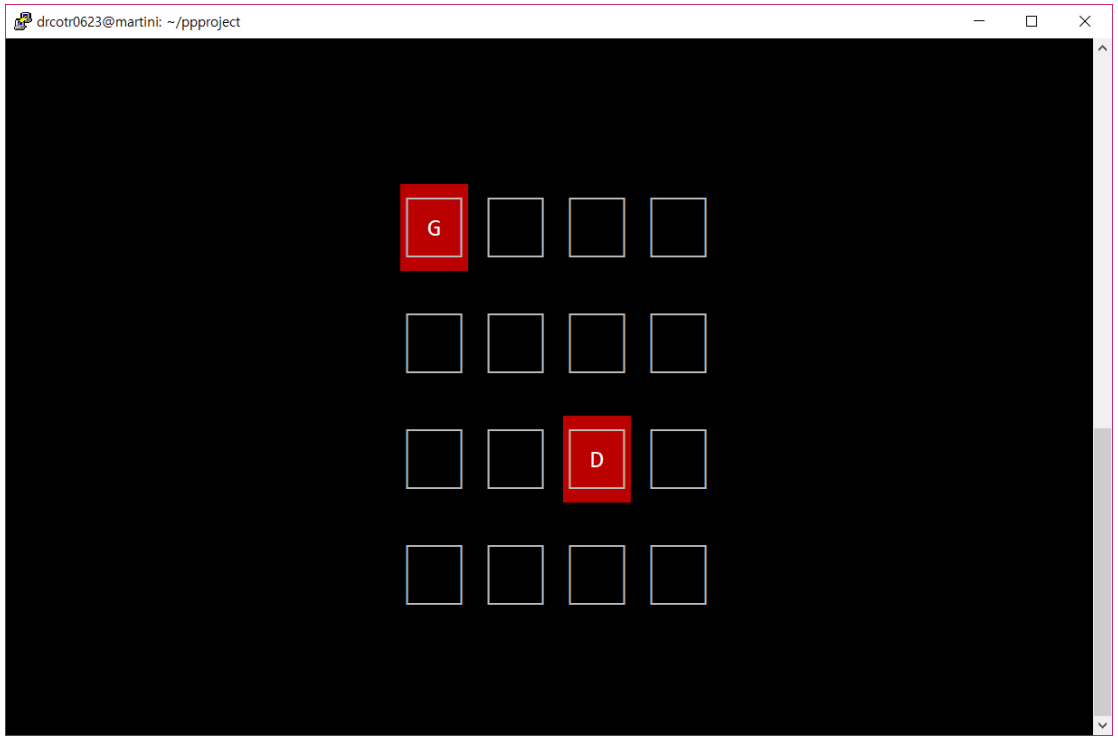
Examples

2. Game Screen



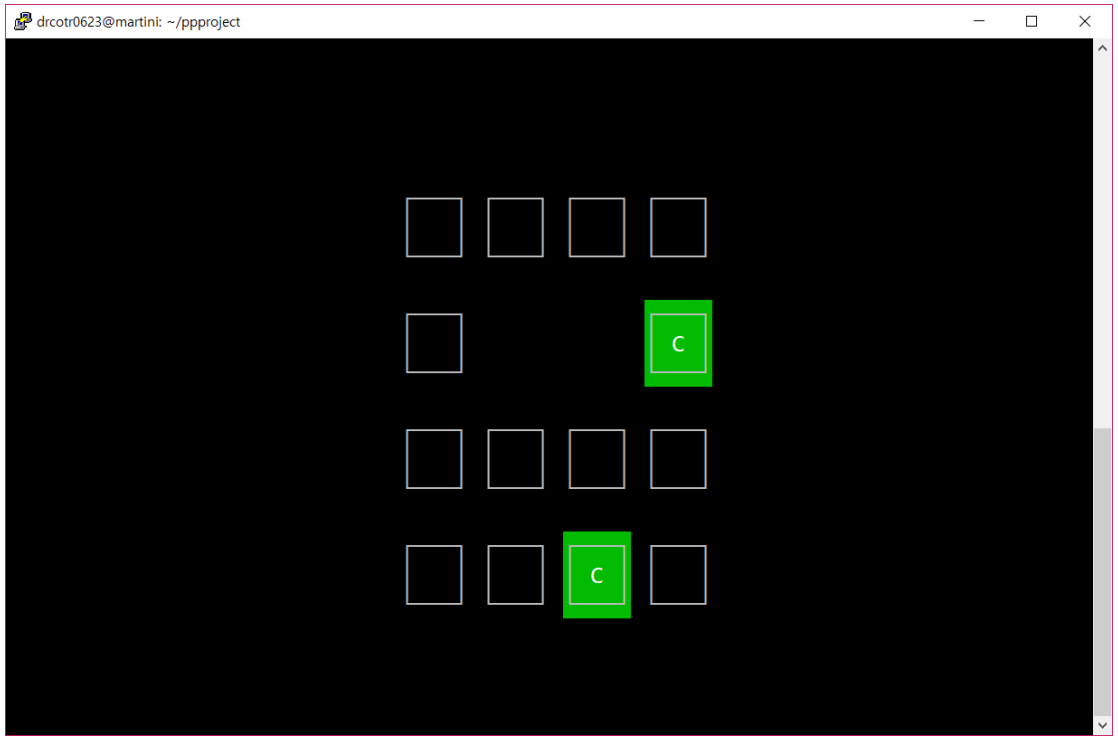
Examples

2. Game Screen



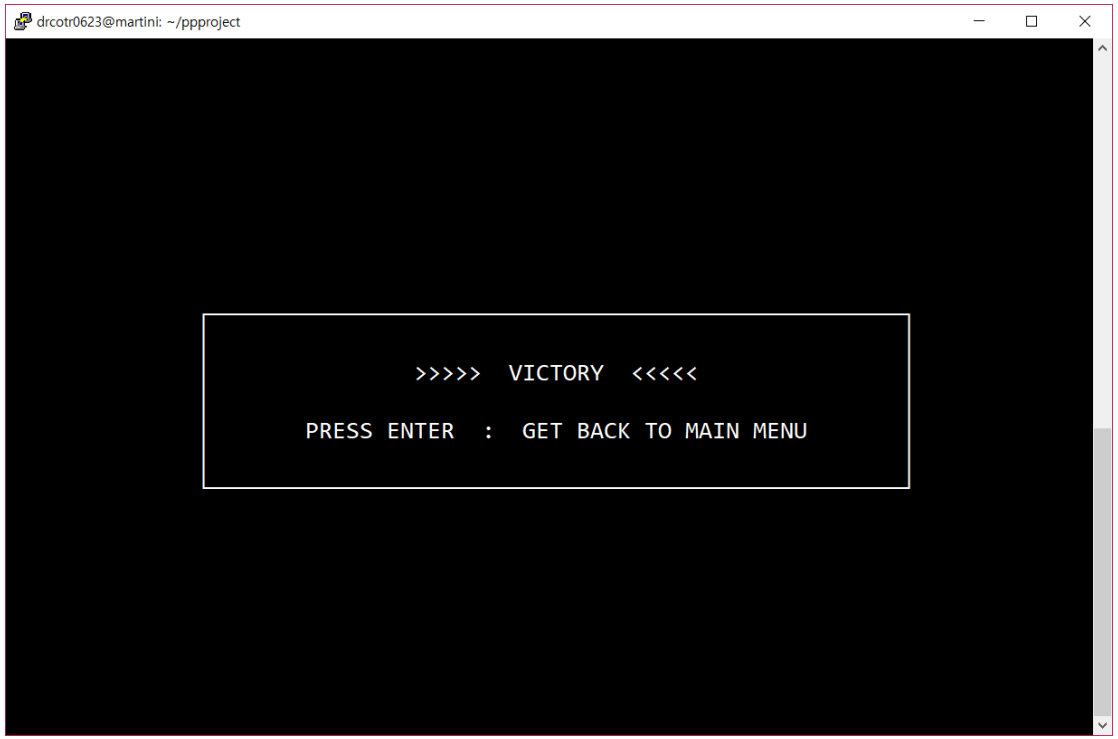
Examples

2. Game Screen



Examples

2. Game Screen



Specific requirements

1. Main Menu (Same as last week)

   - Your game should start from main menu. **(5 pts)**
   - Your main menu should work well. (see last week requirements) **(5 pts)**
     - ‘GAME START’ and 'QUIT' options are necessary.
     - Currently selected option should be highlighted, and it should be possible to select different options using the **<arrow keys>**, and finally choose with **<spacebar>**.
     - When 'QUIT' is chosen, quit the game (exit the program). The program should not exit in any other circumstance (other than when QUIT is entered from the Main Menu). When 'GAME START' is chosen, go to in-game screen.

Specific requirements

2. In-game screen

- Display a grid of 4 x 4 cards. At first, every character of the cards should be hidden. **(5 pts)**
- Random card distribution **(total 10 pts)**
  - Each card must be assigned to exactly one characters. Each character should appear exactly twice. **(2 pts)**
  - Assignment of characters should be random. It should be different for every game. **(5 pts)**
  - It also have to be different whenever you re-execute the program (See Tip 1) **(3 pts)**
- A single card should be highlighted to indicate it is being selected. You also have to be able to move it. (Last week) **(10 pts)**

Specific requirements

2. In-game screen

- It should be able to choose the currently highlighted card with **<spacebar>**. Your chosen card should be highlighted with different color. The character of card must not be disclosed. **(5 pts)**
- You should be able to cancel the selection of card by re-selecting it. **(5 pts)**
- When you have chosen two cards, display the characters of them for a moment.

  (about 0.5~2 sec) **(10 pts)**
- Chosen cards should be removed from the game board if they are identical.

  Otherwise, return it back. **(10 pts)**

Specific requirements

2. In-game screen – Pause Menu & Victory

- Pressing 'q' at any point in the 'in-game' screen should display pause menu. **(5 pts)**
- Displayed pause menu should have three options. **(total 15 pts)**
  - resume by ‘q’ or ‘Q’ **(5 pts)**
  - restart game by ‘r’ or ‘R’ **(5 pts)**
  - return to main menu by **<enter>**. **(5 pts)**
  - You can design your own pause menu.
- After you remove every card, you should display Victory screen. **(10 pts)**
- Pressing **<enter>** at the victory screen, you should go back to main menu. **(5 pts)**

Specific requirements

3. Additional Implement **(extra 10 pts)**
   - Any additional implement will get points according to its difficulty and completeness.
   - However, any bugs for assigned requirement due to additional implementation will also make deduction. **Be Careful!**

Grading policy

- Other specific implementation and designs are all up to you.
  - Ex) Color of highlight, box size, word-phrasing, etc.
- However, bugs that go against common-sense will be deducted.
  - Ex) More than 1 option/card is highlighted at the same time.
  - Ex) Some cards don't highlight correctly. / Game crashes when trying to move the highlight past the right-end or bottom-end, etc.
  - Ex) Screen goes blank for no reason. / Game just randomly quits itself.
  - etc.
- Also, make sure that **`endwin()`** is called before exiting **`main()`** function. (Points will be deducted if this requirement is not met.)

Grading Environment

- Lab computer & Martini server
  (Code should compile correctly and run in at least one of these environments.)

- Compile command:

```
gcc <yourfile.c> -o game –O2 –lm -lncurses
```

Tips

For this week's implementation

Tip1: Generate Random Number

```
#include <stdlib.h>
#include <time.h>

srand(time(NULL));          // Initialize rand() function.

int x = rand();             // Usage : Generate Random Number
```

Tip1: Generate Random Number

```
int rand()


// returns a value from the beginning of random table
// random table is always same if you don’t change it
// → it will return same value whenever you re-execute the
// program
// range of values in the table : 0 ~ 32767
```

Tip1: Generate Random Number

```
time(NULL)

// returns current time in seconds
```

Tip1: Generate Random Number

```
void srand(unsigned int seed)

// change the random table by given seed
// default seed is 1

// use time(NULL) as a seed
```

Tip2: How to Wait for N miliseconds

Implement this function in your code and use it while displaying game board.

```
void milsleep(int ms)
{
        struct timespec ts;
        ts.tv_sec = ms/1000;
        ts.tv_nsec = (ms % 1000) * 1000000;
        nanosleep(&ts, NULL);
}
```