

SEEM3460 Project (2025)

”Higher-Or-Lower” Game

Due on December 1st, 17:00, 2025

1 General Information

- You must follow the guidelines in this file; otherwise, a mark penalty will be applied.
- Development and running of the programs must be on the **Linux platform**

2 Faculty Guideline for Plagiarism

Plagiarism includes copying the whole or parts of programming assignments from others’ work or the Web. A very heavy punishment will be imposed on both the copier and the one whose work is copied.

3 Problem Overview

- **Objective:** Complete the ”Higher-Or-Lower” game in C by implementing required methods, with an optional **EXTENSION PART** for program extension.
- **Key requirements:**

1. The GTK library is required; only linux03-linux04 can compile the programs.
2. Project materials are available in `seem3460/distribute/proj1`.
3. Download commands:

```
cp -r ~seem3460/distribute/proj1 /proj1
cd /proj1
```

4. Compile command (warnings may appear; fix via Task requirements): `make main`
5. Run command (initial window has no functionality): `./main`

A finished program example (WITHOUT EXTENSION) is prepared to show you the game logic:

1. Login to Linux account with X11 Forward enabled.
2. `cp -r ~seem3460/distribute/proj1_demo /proj1_demo`
3. `cd /proj1_demo`
4. `./main`

Note: Ensure X11 Forward is enabled, and VcXsrv (Windows) or XQuartz (MacOS) is installed/opened if no window appears.

4 Problem Specification

4.1 The ”Higher-Or-Lower” Game

- 2 human players, 9 rounds total.
- Game process:

1. Each player gets 1 initial card at the start.
 2. Per round: Player guesses if the next card is "Higher" or "Lower" than the opponent's last card (via GUI buttons).
 3. Scoring: Correct guess = +10 points; Wrong guess = -5 points (all start at 0).
 4. Winning: Player with the highest score after 9 rounds wins.
- Button functions:
 1. "New Game": Reset all and start a new game.
 2. "Quit Game": Close the game window.

4.2 Program Structure

3 provided files (**DO NOT** modify `main.c` unless necessary):

File	Function	Functionality
<code>main.c</code>	<code>main</code>	<code>int main(int argc, char **argv):</code> Main function of the program.
<code>highlow.c</code>	<code>add_new_card</code>	<code>int add_new_card(int container_id, int step):</code> Add specific card image to container (0 = north, 1 = south; step = card index in deck).
	<code>get_prompt</code>	<code>char * get_prompt():</code> Return prompt showing each player's score.
	<code>new_game</code>	<code>void new_game():</code> Create new game (initialize status, clear containers, set prompt, shuffle deck, add initial cards, show ingame buttons).
	<code>card_shuffle</code>	<code>void card_shuffle():</code> Shuffle cards in <code>card_deck</code> .
	<code>end_game</code>	<code>void end_game():</code> Hide ingame buttons and print final winner.
	<code>higher_lower</code>	<code>void higher_lower(int is_higher):</code> Game logic for "Higher/Lower" clicks (draw new card, calculate score, update status).
	<code>on_click_higher</code>	<code>void on_click_higher():</code> Bound to "Higher" button; calls <code>higher_lower</code> .
	<code>on_click_lower</code>	<code>void on_click_lower():</code> Bound to "Lower" button; calls <code>higher_lower</code> .
	<code>on_click_hint</code>	<code>void on_click_hint():</code> (Extension) Show remaining card count.
<code>gui.c</code>	<code>clear_child</code>	<code>void clear_child(GtkWidget* container):</code> Clear all images in the given container.
	<code>clear_container</code>	<code>void clear_container(int container_id):</code> Clear all images by container ID.
	<code>set_prompt</code>	<code>void set_prompt(char *):</code> Display given string as GUI prompt.
	<code>show_ingame_buttons</code>	<code>void show_ingame_buttons():</code> Show "Higher", "Lower", and "Hint" buttons.
	<code>hide_ingame_buttons</code>	<code>void hide_ingame_buttons():</code> Hide ingame buttons.
	<code>get_image_path</code>	<code>char* get_image_path(int card):</code> Given a card ID, generate an image path.
	<code>add_image</code>	<code>void add_image(int container_id, int card):</code> Add card to specified container.
	<code>quit_game</code>	<code>void quit_game(GtkWindow *window):</code> Quit game (used by GTK).
	<code>activate</code>	<code>void activate(GtkApplication *app, gpointer user_data):</code> Register all GUI components.

5 The Task

5.1 Main Part (90% of Total Mark)

1. Design a C program: Create header file `gui.h` for `gui.c`, revise declarations in `main.c` and `gui.c`, and update the makefile for compilation.
2. Implement the following functions to ensure smooth game operation:

- `get_image_path(int card)`
- `quit_game(GtkWindow *window)`
- `end_game()`
- `new_game()`
- `higher_lower(int is_higher)`

3. Prepare a Sample Linux Interaction Output.

Note: Modify only parts labeled "CODE HERE" unless necessary.

5.2 Extension Part (10% of Total Mark)

- Open-ended task: Upgrade the program while preserving core game meaning (otherwise, Main Part marks may be deducted).
- Marks are awarded based on extension difficulty (up to 10% total, **sum of multiple extensions**).
- Required: Submit a report (`<student-ID>-extension-work.pdf`) with **no more than 1 page**, describing extensions and new components.

Extension Options & Marks

- **2%:** Alert user on "Quit Game" (confirm before quitting).
- **5% each:**
 1. Graphical highlight for current player or score status.
 2. Add "Hint" button (show remaining card count).
 3. Add "Cheat" button (display next card).
 4. Add "Pass" button (no guess, 0 points for the round).
 5. Implement a computer player (South).
- **10% each:**
 1. Modify game design/interface for 3 players.
 2. Add "Back" button (retract 1+ steps).

5.3 Sample Linux Interaction Output

The Linux command `script <sample-output-filename>` should be used to capture the terminal data and information. Once the `script` command is issued, it will set up an environment. From this moment, users can continue to use Linux. Whatever the user types and information displayed on the terminal will be automatically saved into the `<sample-output-filename>`, which will be treated as the Sample Linux Interaction Output for later submission. (Use CTRL-D to quit the terminal capture.) The file content **MUST** contain the following:

1. Display the content of the makefile by the command `cat makefile`.
2. Compile the program by the command `make`.
3. Execute the executable file by the command `./main`.

After capturing the content of the "Sample Linux Interaction Output" file, copy a version of this file with the name `<student-ID>-project-interact.txt`.

6 Submission

6.1 Required Files

1. `main.c`
2. `highlow.c`
3. `gui.c`
4. `gui.h`
5. `makefile`
6. `<student-ID>-project-interact.txt`
7. `<student-ID>-extension-work.pdf` (only if Extension Part is completed)

6.2 Submission Steps

1. Download files 1-6 from the Linux server to your local machine.
2. Place all files (1-7) in a folder named `<student-ID>-project`.
3. Compress the folder into `<student-ID>-project.zip`.
4. Submit the zip file to CUHK Blackboard (multiple attempts allowed; last submission is graded).

7 Question & Answer

1. First check the Q&A Google Document for existing answers: **click here**.
2. Add new questions **that have not been asked yet** to the Google Document, we will answer you there when we see your questions.
3. Still have questions or meet with problems: Send emails to TAs.