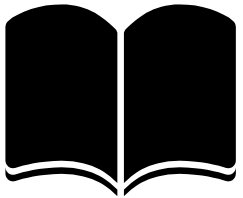
SEEM 3460
Mini-Project
“Higher-Or-Lower”

Yang DENG
ydeng@se.cuhk.edu.hk

Environment Configuration



Windows

VcXsrv: https://sourceforge.net/projects/vcxsrv

If you need to run GUI programs.
run putty.exe

- [Connection]-[SSH]-[X11]:
  - **Enable X11 forwarding**
- [Session]:
  - Host Name: (e.g.) cuse81
  - **Save** for later use
  - **Double click** the saved session name to start SSH

MacOS

Xquartz: https://www.xquartz.org

ssh –X user@linux03.se.cuhk.edu.hk

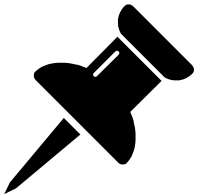
(Same as Assignment 2)

1.

Project Overview

- The objective of this assignment is to complete the game “**Higher-Or-Lower**” in **C** by implementing several methods.
- The project also has an **EXTENSION PART** to let you extend the program in your own way.

1) The GTK library is required for this project. Currently only linux03-linux05 can compile your programs.
2) Material for this project can be found in this folder:
   ~seem3460/distribute/proj1
3) To download the files to your own home directory, run the following commands:
   cp -r ~seem3460/distribute/proj1 ~/proj1
   cd ~/proj1
4) Compile the program with the following commands (You may see a lot of warnings. Please see the Task part to fix them):
   make main
5) Run the program with the following commands (You will see a window but it can do nothing):
   ./main



Demo

1) cp -r ~seem3460/distribute/proj1_demo ~/proj1_demo
2) cd ~/proj1_demo
3) ./main

(If you cannot see any windows open, please ensure you have enabled the X11 Forward and VcXsrv is installed and opened. Or XQuartz in MacOS)

Problem Specification

2.

- The “Higher-Or-Lower” Game
- Program Structure
- The Task - Main Part
- The Task - Extension Part

“Higher-Or-Lower”

1) The game is played by 2 human players. At the beginning of a new game, **one card** is dealt to each player.
2) The game has **9 rounds**. In each round, an extra card will be dealt to each player.
3) Before the card is dealt, the player guesses whether the coming card is higher or lower than his opponent’s last card dealt by clicking the “**Higher**” or “**Lower**” button.
4) A correct guess **wins 10 points** while a wrong guess **loses 5 points**.
5) All players start from zero points. After the rounds, the player who scores most points wins.
6) The “**New Game**” button, whenever it is clicked, should reset everything and start a new game. The “**Quit Game**” button should close the game window.

Program Structure

1) Some C functions in some files are provided to you as a starting point. It contains 3 files: **main.c**, **gui.c** and **highlow.c**.
2) DO NOT modify anything in **main.c** unless you have particular demand.

File	Function	Functionality
main.c	main	int main (int argc,char **argv). The main function for our program. Please **DO NOT** modify this file.

File	Function	Functionality
highlow.c	add_new_card	int add_new_card(int container_id, int step); Add specific card image into a container. If container = 0, it will add card into the north container. If container = 1, it will add card into the south container. Step indicates which card in the deck will be used.
	get_prompt	char * get_prompt(); Get the prompt to show. It will show each player's score.
	new_game	void new_game(); Create a new game. It should 1) initialize the status variable; 2) Clear north and south container; 3) Set the initial prompt; 4) Shuffle the card deck; 5) Add initial card for north and south player; 6) Show ingame buttons.

File	Function	Functionality
highlow.c	card_shuffle	void card_shuffle(); Shuffle the cards in the card_deck.
	end_game	void end_game(); It will 1) Hide the ingame buttons; 2) Print the final results about who win the game.
	higher_lower	void higher_lower(int is_higher); Game logicals for users if they press higher or lower. It will: 1) Draw a new card from the card container; 2) Calculate the score for the player; 3) Update the status variable.
	on_click_higher	void on_click_higher(); Will be bound to GUI button Higher. It will call the higher_lower function.

File	Function	Functionality
highlow.c	on_click_lower	void on_click_lower(); Will be bound to GUI button Lower. It will call the higher_lower function.
	on_click_hint	void on_click_hint(); (**Extension part**) Show how many cards are remaining.
gui.c	clear_child	void clear_child(GtkWidget* container); Clear all images in the container by the given container pointer.
	clear_container	void clear_container(int container_id); Clear all images in the container by the given container ID.

File	Function	Functionality
gui.c	set_prompt	void set_prompt(char *); Show the given string in the GUI as the prompt.
	show_ingame_buttons	void show_ingame_buttons(); Show ingame buttons including Higher, Lower, and Hint.
	hide_ingame_buttons	void hide_ingame_buttons(); Hide ingame buttons including Higher, Lower, and Hint.
	get_image_path	char* get_image_path(int card); Given a card ID, generate an image path. In most places of the program, cards are indexed using their rank: 0=♣2 < ♦2 < ♥2 < ♠2 < ♣3 < … < ♠A=51 But the PNG files in picture folder uses another index. So we need a function for conversion.

File	Function	Functionality
gui.c	add_image	void add_image(int container_id, int card); Add the card into the container.
	quit_game	void quit_game(GtkWindow *window); Quit the game. This will be used by GTK.
	activate	void activate (GtkApplication *app, gpointer user_data); Register all GUI components.

Task - Main Part

The followings are the main tasks:

1) To implement a good C program design, create header file **gui.h** for **gui.c** and revise the declaration in **main.c** and **gui.c** accordingly. Revise the **makefile** to compile the program.
2) Complete **get_image_path (int card)**, **quit_game(GtkWindow *window)**, **end_game()**, **new_game()**, **higher_lower(int is_higher)**, to make the game run smoothly.
3) Prepare a **sample Linux interaction output**.

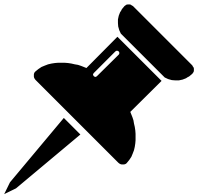
The **MAIN PART** constitutes **90%** of the total mark. You are strongly advised not to modify any part except parts labelled with the "**CODE HERE**" comments. But the **EXTENSION PART** in general requires some other extra modification.

Sample Linux Interaction Output

The Sample Linux Interaction Output should contain some sample Linux input commands and 5 output for task management and compilation. It also contains sample execution of the program.

- The Linux command “**script** ” should be used to capture the terminal data and information. Whatever the user types and information displayed on the terminal will be automatically saved into the **<sample-output-file>**. (Use **CTRL-D** to quit the terminal capture.)
- The file content MUST contain the followings:
  - ☐ Display the content of the makefile by the command "**cat makefile**"
  - ☐ Compile the program by the command "**make**"
  - ☐ 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**

Extension Part

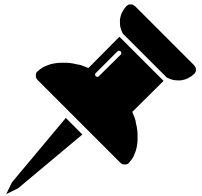


- The extension part is an open-ended task that constitutes **10%** of the total mark.
- You may do any kind of upgrade to the program given that your final program **keeps the meaning of the game well** (otherwise marks will be deducted for your MAIN PART). Marks will be awarded according to the difficulty of the extension you made.
- You should also write **a report not exceeding one page**.
- The awarded marks will be up to the SUM OF VALUE of simultaneous extensions you made, bounded by 10%.

Extension Part

- The following extension is worthy of **2%**:
  - ☐ **Alert user when user clicks “Quit Game”**: ask the user if she/he would like to quit the game. If the user clicks “Yes”, then quit. Otherwise, keep the status
- Each of the following extension is worthy of **5%**:
  - ☐ **Make a graphical highlight** to show either: the player in turn; or the score status
  - ☐ **Add a “Hint” button** for showing the count of remaining cards
  - ☐ **Add a “Cheat” button** for displaying the next card in the game interface
  - ☐ **Add a “Pass” button** for not making a guess and gets zero point in a round
  - ☐ **Implement a computer player** that plays South
- Each of the following extension is worthy of 10%:
  - ☐ Change the game design (including the interface) to make it a **three-player game**
  - ☐ **Add a “Back” button** for retracting one or more steps

Extension Part



Note:

- If you complete the extension part, you should write a report **not exceeding one page**.
- The file name should be **<student-ID>-extension-work.pdf**
- The report should describe what you have done in the extension part and the meaning of each new class.

Submission

3.

- There are seven files that should be contained in the submission, including (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
- Download the files (1-6) from the Linux server to your local machine (same as Assignment 2).
- Place all these files and your report (7) in the same folder **<student-ID>-project**
- Compress the folder **<student-ID>-project** into one zipped file **<student-ID>-project.zip**
- Submit the zipped file to the CUHK Blackboard. Multiple attempts are allowed before the deadline but the last attempt will be graded.

Question & Answer

4.

- With any questions about the mini-project, please check the following Google Document for Q&A first and write only new questions that have not been asked yet:

  https://docs.google.com/document/d/1ELPYct1N0npCJNIME4JtxnX8-JjDrekO2QGRJ8LkgCA/edit?usp=sharing

- Or send an email to ydeng@se.cuhk.edu.hk

Thanks!

Any questions?