**Bug Report:**

During the project demonstration, my program had following bugs:

- Did not print the piles of the players when a round was over. I fixed the bug immediately after the demonstration. The project does not have this bug anymore

- The program correctly saves the state of the program. All data including table, hand, pile, round number and score is properly saved. The program also correctly loads from a serialization file. However, the game stores information in a slightly different manner than the format of serialization files we used in the demonstration. The correct format of writing a build on a table should have been

[C6 S3] . However, my program writes the build in this format [C6 S3 ], which has extra space after the last card in the build. So, when I try to load the saved file, the serialization did not work properly. When I manually opened the file and removed the extra space after the last card in the build, the program worked perfectly.

**Features Report:**

I implemented all the features we were asked to implement in this project. The features work properly and handle edge cases. No missing features. No additional features.

**Description of data structures/ classes**

- **Classes**

I have used eleven classes in the program. They are Card, Deck, Build, MultiBuild, Table, Strategy, Player, Human, Computer, Round, Tournament.

Card – Represents a card that has suite character and a face character

Deck – Has a collection of cards. Dealing shuffling and dealing of cards.

Build – Build is a collection of loose cards.

MultiBuild – Is a collection of single builds

Table -Table has three components – Loose cards, Single Builds and Multi Builds.

Strategy - The computer uses this strategy to figure out what kind of move to make and what card to play. Also used to provide help to human player.

Player – A virtual class which holds hand, pile, single builds and multibuilds.

Human and Computer - Inherit from the Player class. Player class is a virtual class as it had several functions for controlling the game that are similar but have different implementations for computer and human. No other inheritance is used.

Round – Plays an entire round of a game. The round class hold Deck of cards, Table, Human player and Computer player.

Tournament – Class plays the entire tournament

main.cpp- Not a class. Simply an entry point for the program

**How to run the program:**

1) Compile only the cpp files, including main.cpp

2) Run the program

3) The program starts running from main.cpp as int main in main.cpp is the entry point for the program

For instance in Linux based systems:

Step1:

Open the terminal and navigate to ghimire/src

Type:

g++ Card.cpp Deck.cpp Build.cpp MultiBuild.cpp Table.cpp Player.cpp Human.cpp Computer.cpp Strategy.cpp Round.cpp Tournament.cpp main.cpp

Step 2: Type ./a.out on the teminal and run the program.

If Step 1 does not work, follow the following steps:

Open the terminal and navigate to ghimire/src

Type:

g++ Card.h Card.cpp Deck.h Deck.cpp Strategy.h Strategy.cpp Build.h Build.cpp MultiBuild.h MultiBuild.cpp Table.h Table.cpp Player.h Player.cpp Human.h Human.cpp Computer.h Computer.cpp Round.h Round.cpp Tournament.h Tournament.cpp main.cpp

Upon doing this, you may see a bunch of errors. Simply ignore them

Type this on the terminal:

g++ Card.cpp Deck.cpp Build.cpp MultiBuild.cpp Table.cpp Player.cpp Human.cpp Computer.cpp Strategy.cpp Round.cpp Tournament.cpp main.cpp

Type ./.a.out and run the program

If you are using Visual Studio or IDEs like that, simply extract file called ghimire into a convenient directory. Then, open the project from the IDE. Build and run

**Log:**

09/09/18

* Worked on understanding the rules of the game. Asked Prof Kumar a question about continuing the round by dealing more cards.
* Prof Kumar said, “If human player has played all the cards, computer will have exactly one card to play”

Wait for the computer to play before dealing cards again

* Figure out what classes to use for the program: At least Card, Deck, Player, Computer, Human, Round, Tournament should be there in the tournament

3 hrs

09/13/18

* Designed, implemented and tested Card and Deck class.
* It was tricky trying to create a deck of unique cards. Decided on what data structure to use to model the deck. Decided to use a queue where the first element in the deck is the first element that is gonna be dealt.
* Stored card objects on the deck
* Used a text file called ‘populateTheDeck’, wrote 52 unique cards on that file and read the file to load individual cards to the deck line by line.
* Still can’t shuffle the deck.

3 hrs

09/15/18

* Changed the data structure to hold cards in Deck. Instead of using a queue, used vector.
* Dynamically created Card objects in populateTheDeck function and added a reference to the Card to the deck. Did this because same card is move around in player’s hand, pile, table and builds. So, creating copy of cards between functions could be error prone and tedious.
* Used random\_shuffle function in Deck’s constructor to shuffle the vector containing the references.

2.5 hrs

09/18/18

* Started working on the Player class, Human class and Computer class.
* A player has a hand (vector of Card\*), a pile(a vector of Card\*), player type (can be either human or computer) and functions like add to pile, remove from pile, add to hand, remove from hand
* Human and Computer classes inherit information about pile and hand from Player class.

4 hrs

* Brainstormed about what a round should have. It can have 2 players, a deck.
* Thought about how to implement table. Decided on creating a table.
* Started working on Build class. A table has loose cards and builds
* Do I need a different class called multi build?

5 hrs

09/18/18

* Started designing Tournament class. Implemented tournament having two players , scoring and determining the winner.
* You need to work on Round before you add working on the Tournament

2 hrs

* Implemented toss and dealing more cards to player (through continueTheRound) until the round is over.
* Removed all scoring related functions from Tournament to round

3 hrs

09/21/18

* Designed how users should play the round.
* Show the table, the players hand before each player plays. Allow the player to play move related options using integers.
* For instance if you want to play a card from the hand, implement a function called something like getCardFromHandIndex . Display an integer next to each loose card, each build, multibuild on table and same from players hand.
* Implemented the functions discussed in the line right above this

4 hrs

09/22/18

* Implemented creating a single build

5 hrs

09/23/18

* Implemented increasing an opponents build, creating multi build.

4 hrs

09/27/18

* Implemented trailing. Made sure that people with build or multi builds cold not trail
* Worked on capture. Allow the user to make different type of captures with one played card.
* If the user has captured any cards, the boolean function capture will be true.
* Similarly if the user has trailed validly, the trail functions is true. Same for build
* IF build || capture || trail is not true, don’t change turn

3 hrs

* Implemented capturing capture single builds, multi builds, sets of loose cards and individual loose cards.

3hrs

09/29/18

* Completed designing a basic two person game.
* Need to implement : Computer Strategy

1.5 hrs

* Thought it would be ideal to implement a class called Strategy that the players can use. Strategy can also be used by round.
* Made the computer extend an opponent’s build. The computer can also trail

3 hrs

10/01/18

* The computer can capture multibuilds, an single builds on the table. However, it still can’t capture sets of loose cards/
* Came out with a recursive algorithm that gets a sets of numbers given a target number.
* This function can be used to create build and capture set of cards

6 hr

10/02/18

* Started working on serialization. Loading the file. Worked on a test program before implementing serialization directly on the project.
* Learned to extract information from file using stringstream and istream objects.
* Can extract human and computer’s data apart from build from the serialization file

10/05/18

* Implemented serialization. However, I got a segmentation fault.
* Spent 1 hour trying to figure out where the segmentation fault was
* Segmentation fault was caused by trying to acess the value of playerWhoCapturedLast, a player pointer wen it was null

3 hrs

10/20/18

* Added a boolean parameter , isAceFourteen with default value of false to getNumericValue of card. If ace is fourteen is true, value of a ce is 14. Else it is 1/
* Implemented the value of ace card being both 14 and 1.
* The ace becomes 14 when it is the hold card for a build, multi-build or for a set of cards (when capturing). Other than that, it is always 1

1 hr

10/21/18

* Implemented hold card. Player class has two multimaps, m\_holdCardsForSingleBuilds and m\_holdCardsForBuildBuilds. Whenever a build is created, a new pair of values is added to the multimap.. The key is the numeric value of the key. Value is the actual Card \*
* In Round class make sure that the user could not play hold cards unless he was capturing his own build or multibuild

3 hrs

10/22/2018

* Properly documented Card, Deck, Player and Human classes

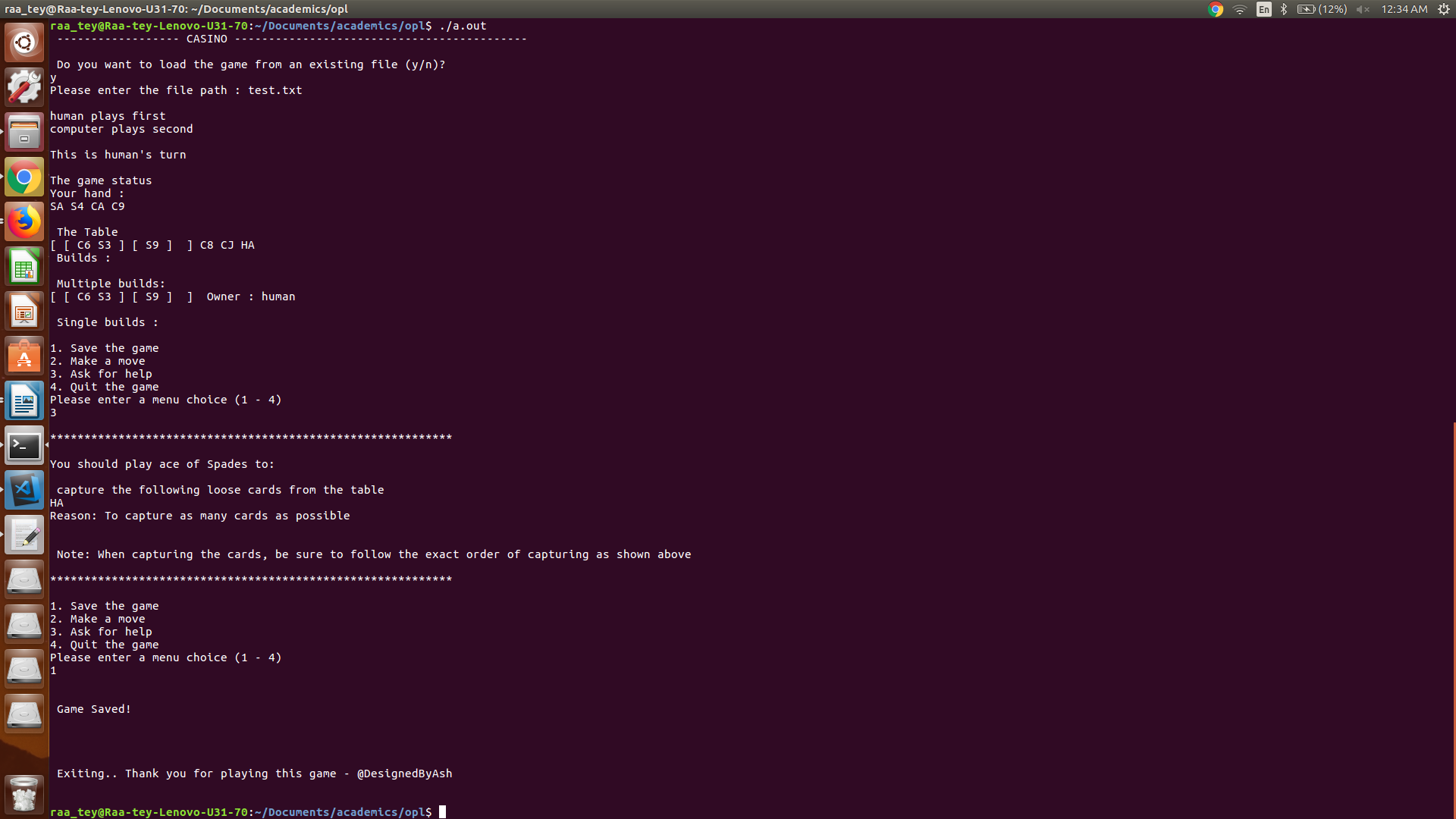
5 hrs

10/23/2018

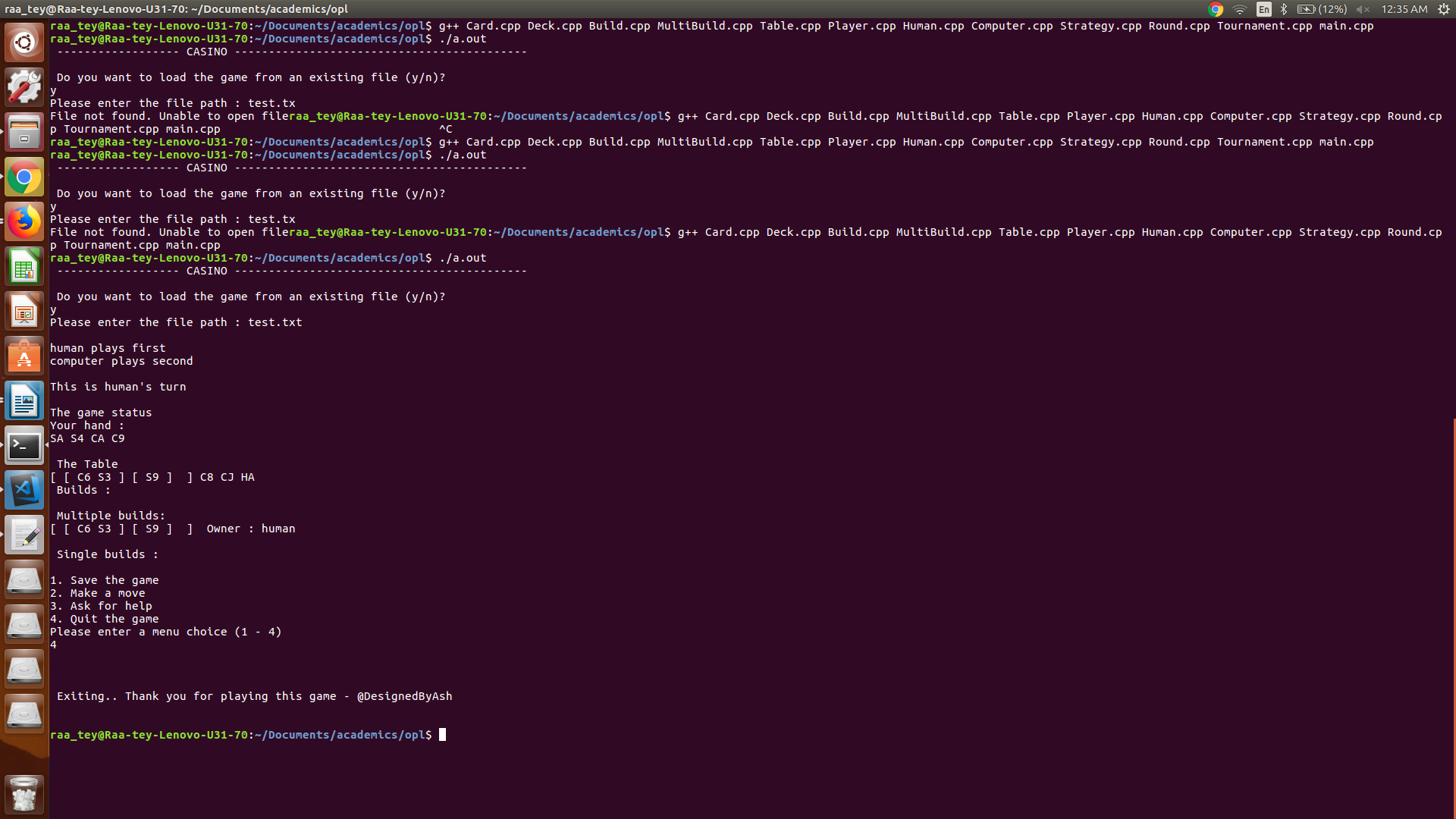
* Documented remaining classes

6 hrs

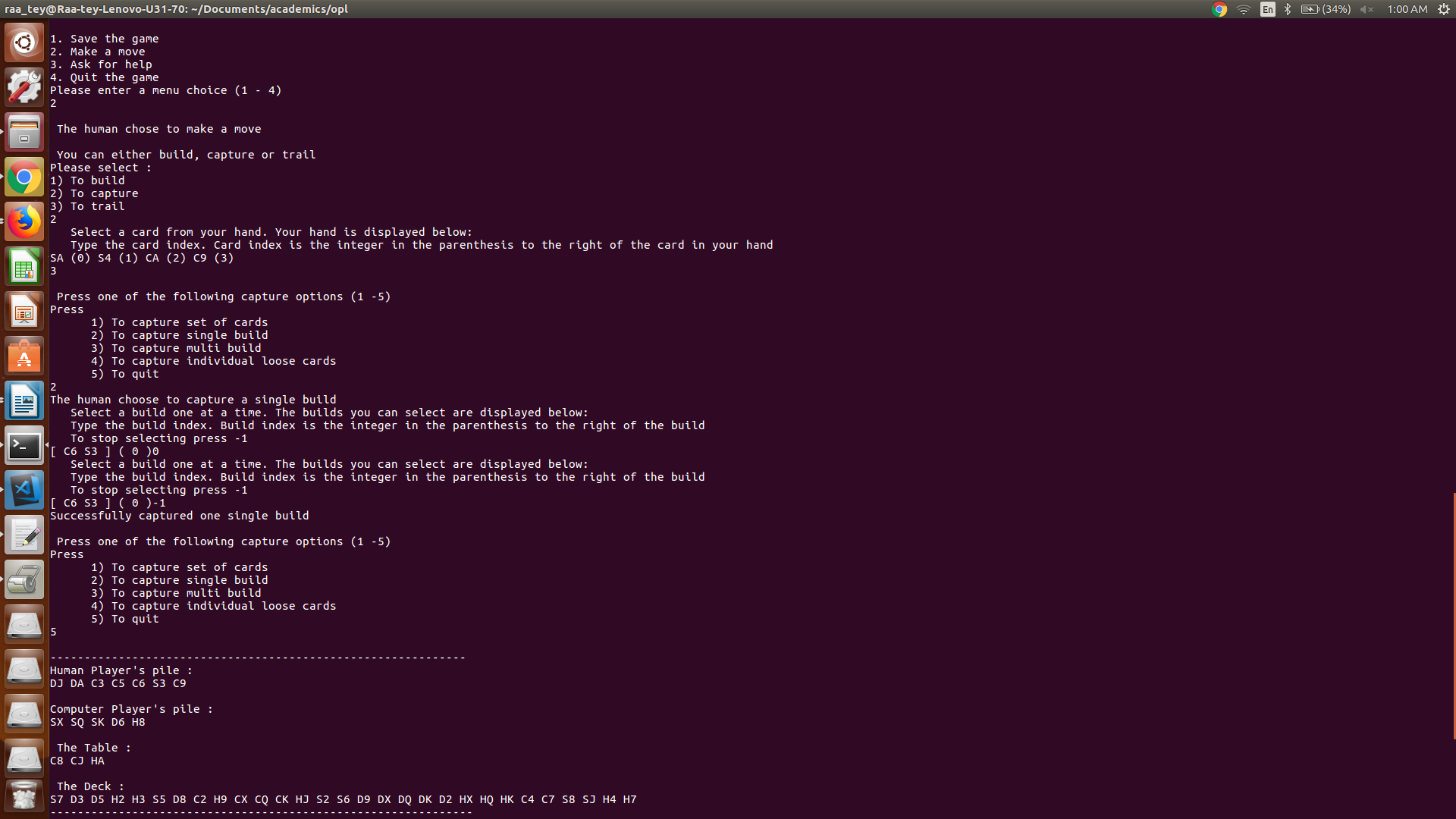
Screenshots



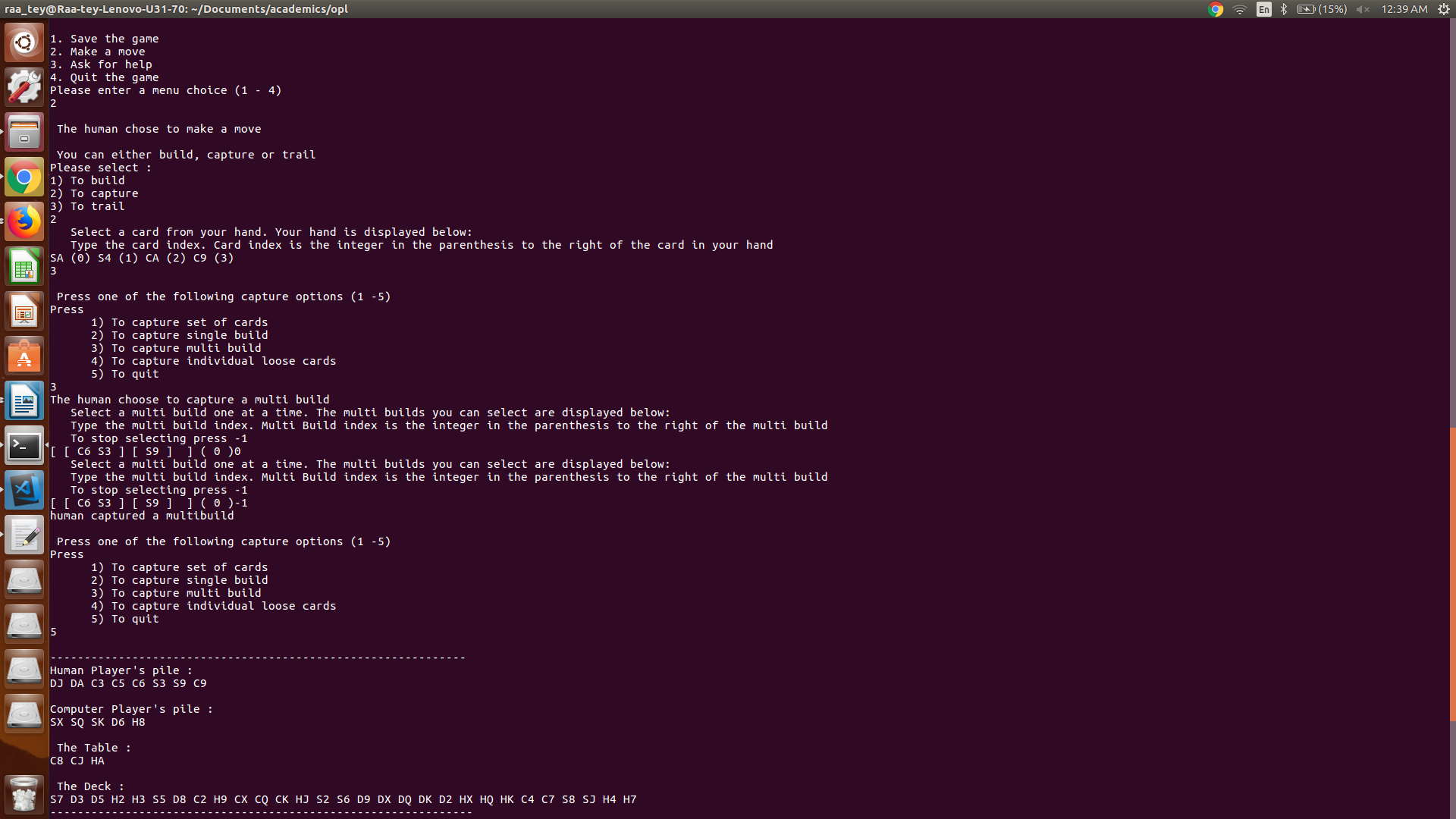
asking for help and saving the game



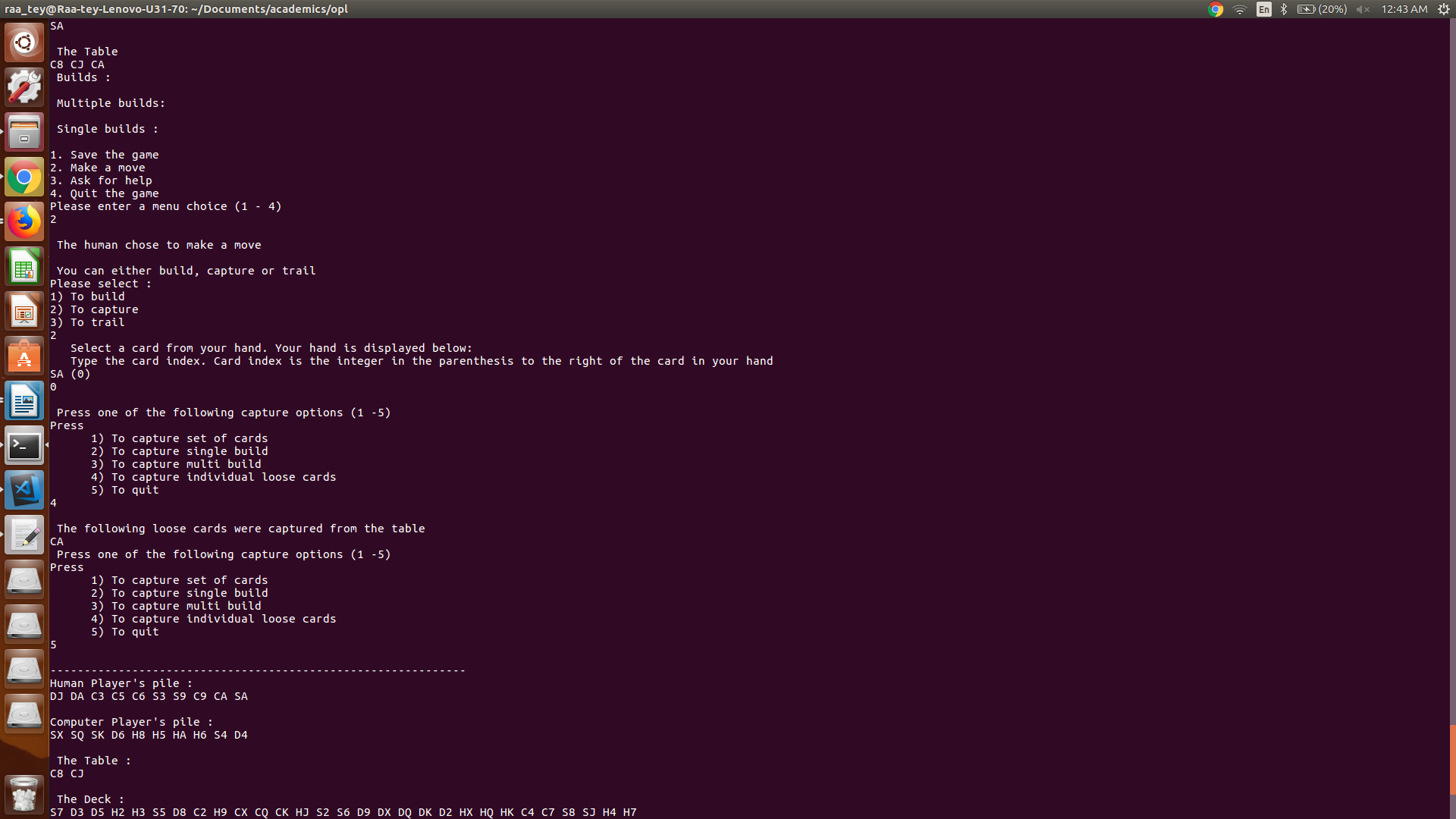
Quitting the game



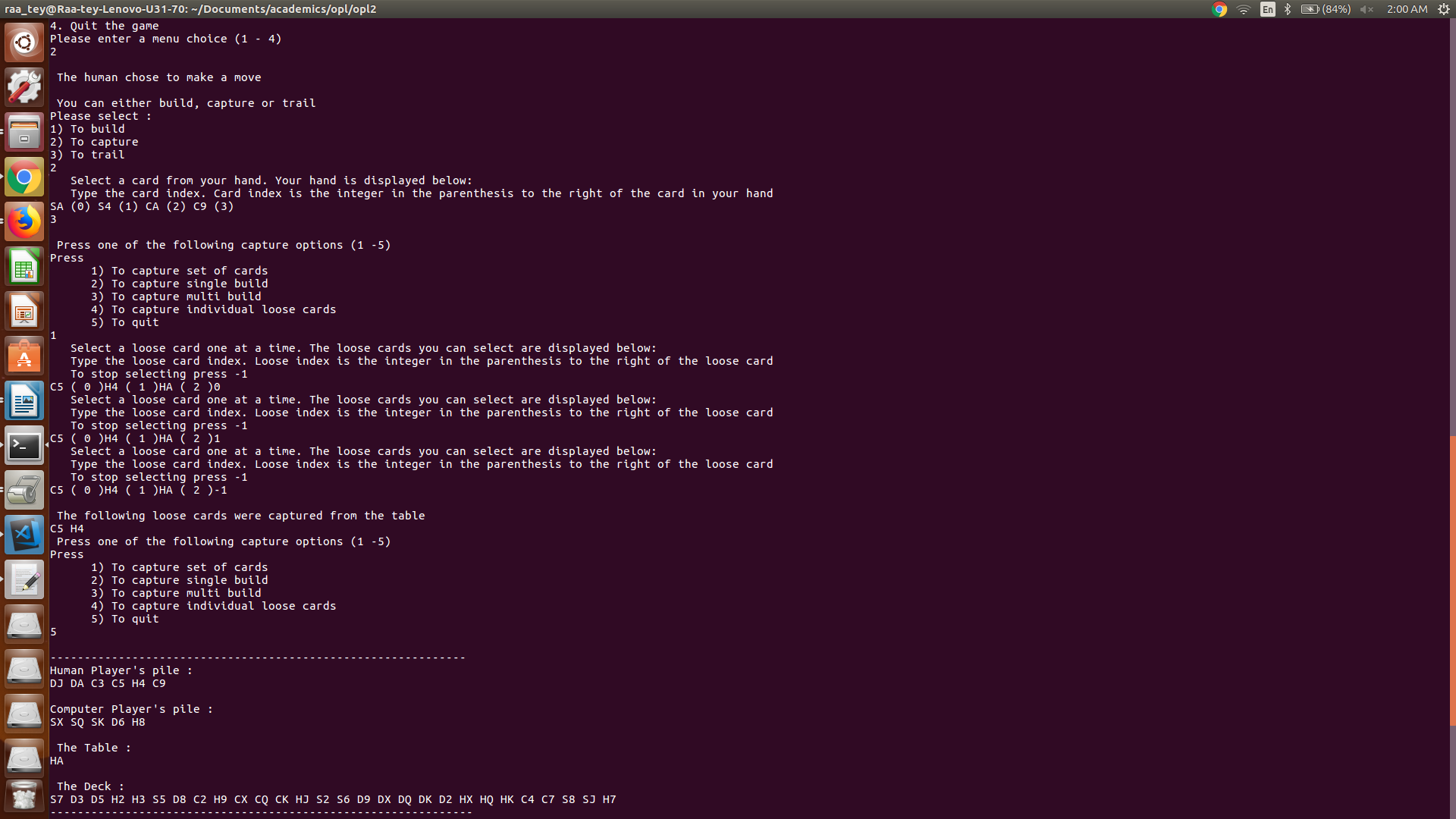
Human capturing single build



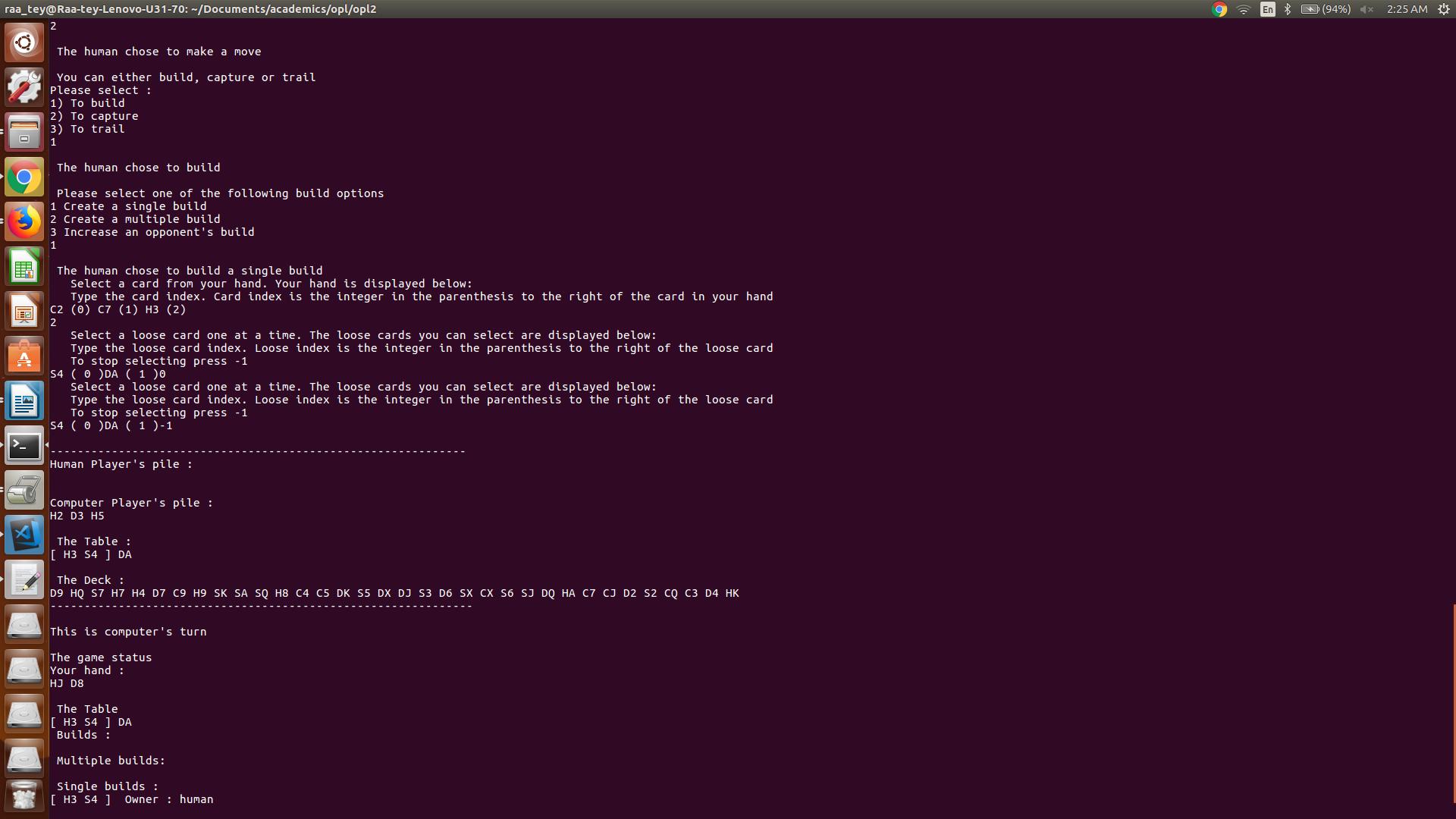
Human capturing a multi build



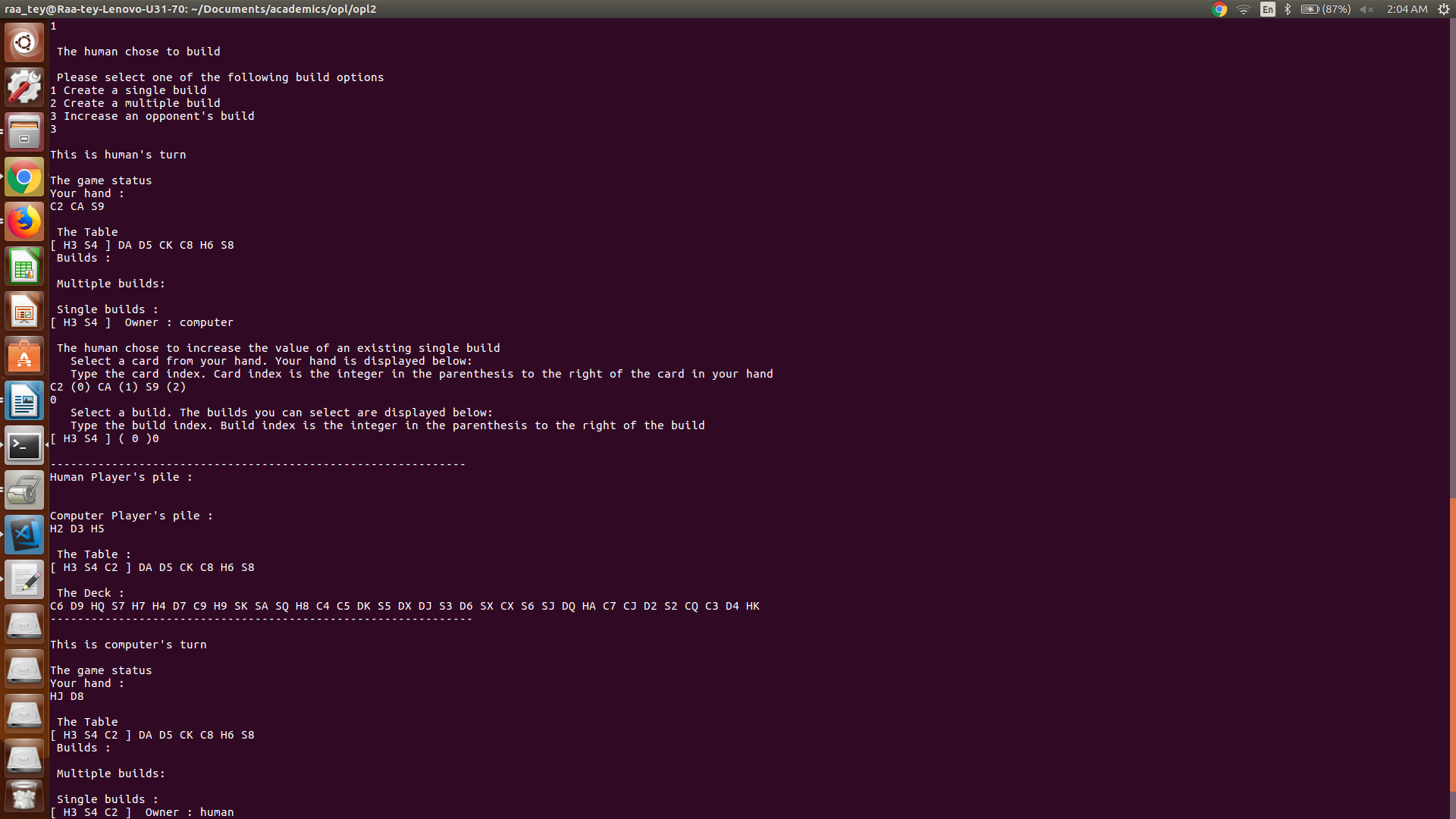
Human capturing loose card



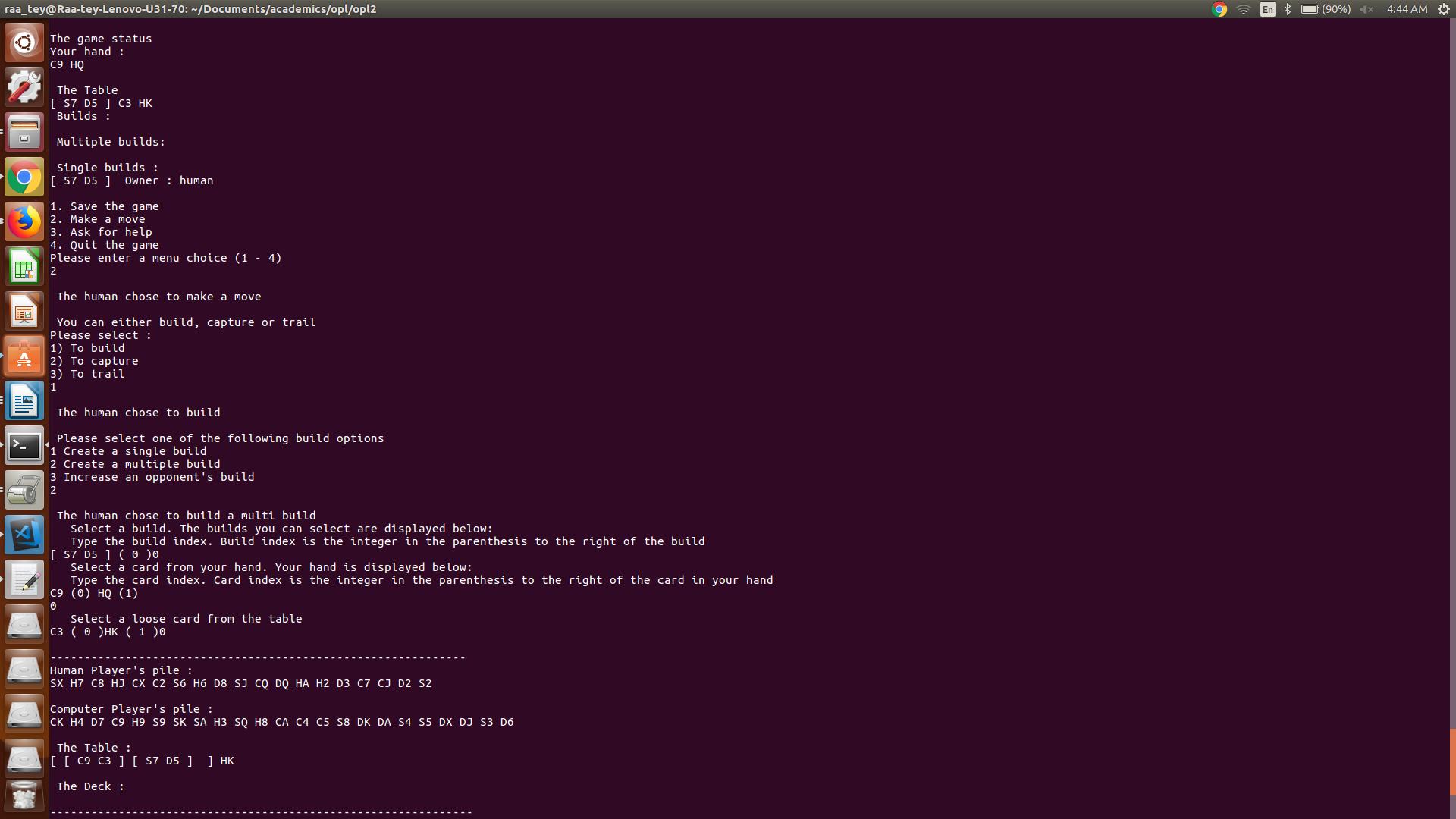
Human capturing set of loose cards



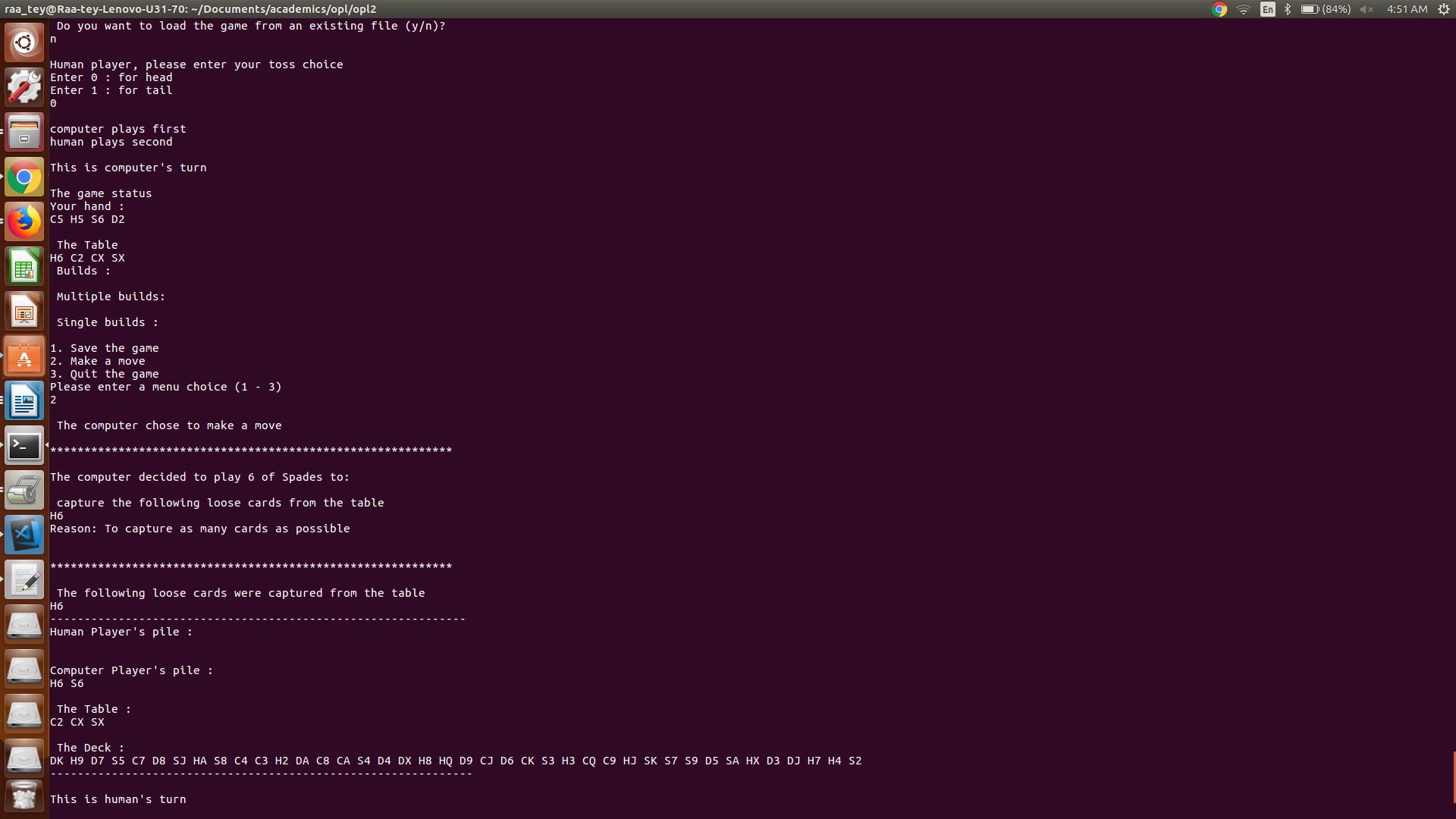
Human creating a single build



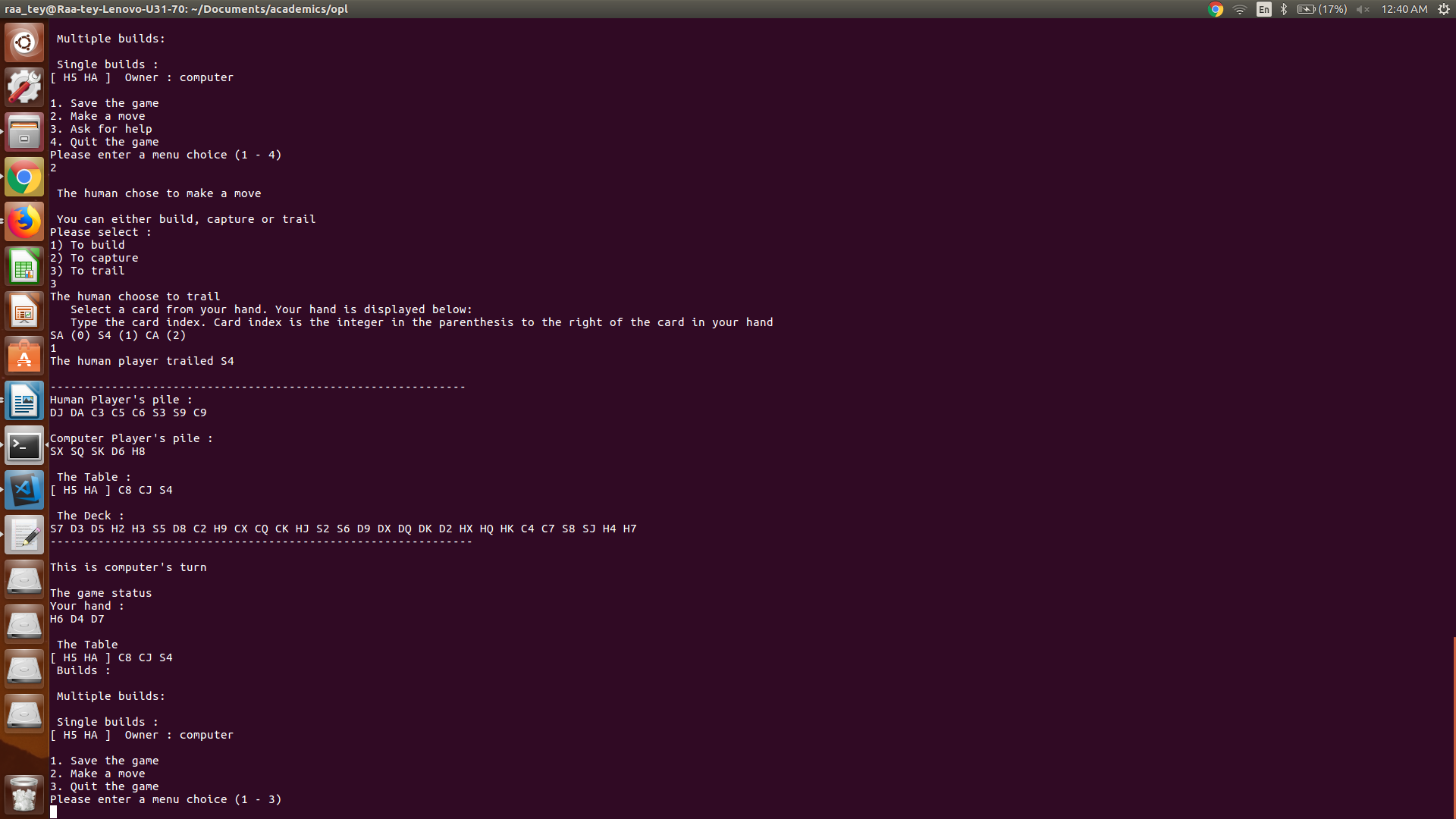
Human extending an opponent’s build



Human creating a multi-build



Overall layout



Human trailing