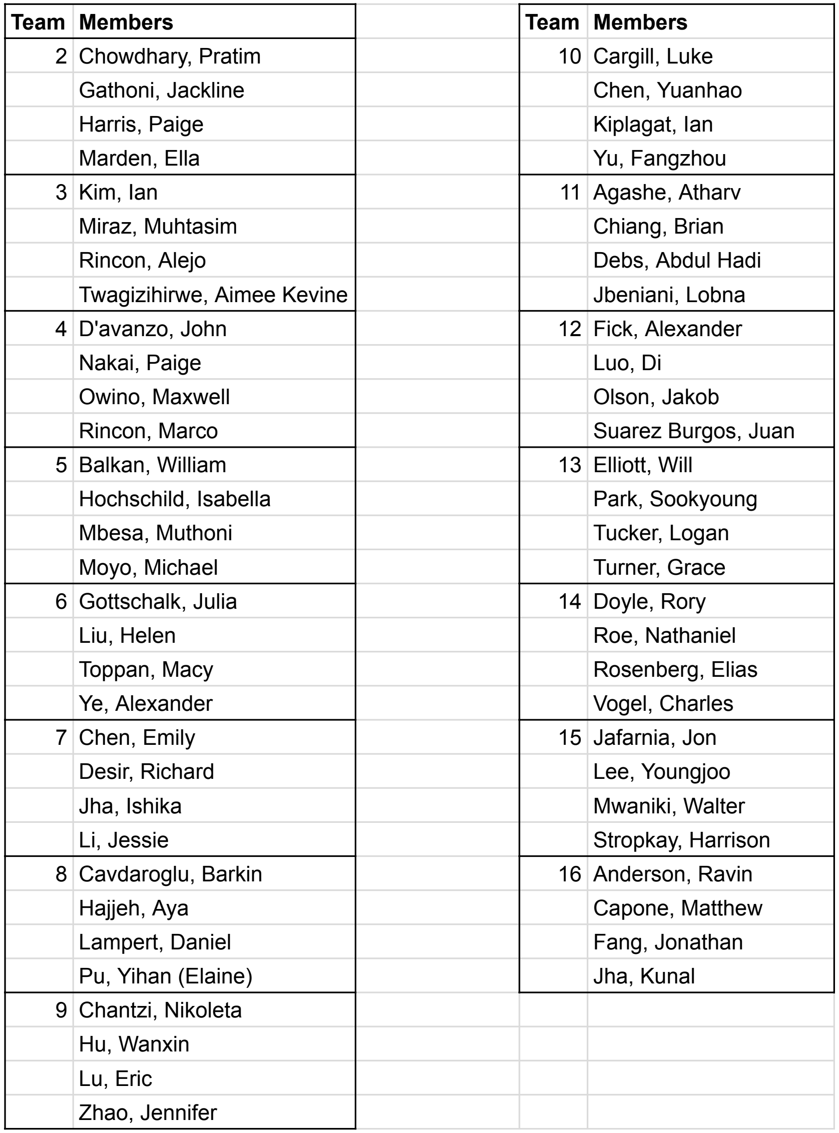
CS 50:
Software Design and Implementation

Final Project

Agenda

1. Project details
2. Implementation
3. Grading
4. Tips

Project teams are on Canvas under Pages



Team	Members
2	Chowdhary, Pratim
	Gathoni, Jackline
	Harris, Paige
	Marden, Ella
3	Kim, Ian
	Miraz, Muhtasim
	Rincon, Alejo
	Twagizihirwe, Aimee Kevine
4	D'avanzo, John
	Nakai, Paige
	Owino, Maxwell
	Rincon, Marco
5	Balkan, William
	Hochschild, Isabella
	Mbesa, Muthoni
	Moyo, Michael
6	Gottschalk, Julia
	Liu, Helen
	Toppan, Macy
	Ye, Alexander
7	Chen, Emily
	Desir, Richard
	Jha, Ishika
	Li, Jessie
8	Cavdaroglu, Barkin
	Hajjeh, Aya
	Lampert, Daniel
	Pu, Yihan (Elaine)
9	Chantzi, Nikoleta
	Hu, Wanxin
	Lu, Eric
	Zhao, Jennifer

Team	Members
10	Cargill, Luke
	Chen, Yuanhao
	Kiplagat, Ian
	Yu, Fangzhou
11	Agashe, Atharv
	Chiang, Brian
	Debs, Abdul Hadi
	Jbeniani, Lobna
12	Fick, Alexander
	Luo, Di
	Olson, Jakob
	Suarez Burgos, Juan
13	Elliott, Will
	Park, Sookyoung
	Tucker, Logan
	Turner, Grace
14	Doyle, Rory
	Roe, Nathaniel
	Rosenberg, Elias
	Vogel, Charles
15	Jafarnia, Jon
	Lee, Youngjoo
	Mwaniki, Walter
	Stropkay, Harrison
16	Anderson, Ravin
	Capone, Matthew
	Fang, Jonathan
	Jha, Kunal

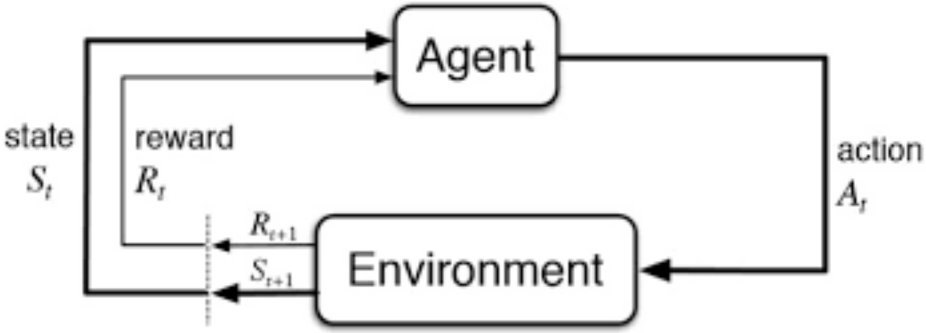
Canvas will have the latest if there are any changes!

Pick one person to accept the assignment, then grant others access to GitHub

Use reinforcement learning to play the game of 21

Reinforcement learning

- Machine learning technique
- Agent:
  - Observes the state of the environment
  - Takes an action based on its observation
  - Receives a reward based on its action
- Explore (training) phase:
  - Try random actions in each state
  - Keep track of average reward for each state
- Exploit (playing) phase:
  - Choose the action with the highest average reward in each state



21 (aka blackjack) is played with a dealer and one or more players

21

- Player initially dealt two cards face up
- Dealer dealt one card face down, one card face up
- Objective: get as close to 21 points as possible without going over
- Points are based on cards:
  - Numbered cards (2-10) points are same as number
  - Face cards (Jack, Queen, King) are worth 10 points
  - Ace is worth either 1 or 11 points (can change if more favorable)
- Sum points for each card to get total points
- Player actions:
  - HIT: take another card
  - STAND: stop taking cards
- Dealer actions:
  - Takes cards after all players finish
  - HITs until 17 points or more

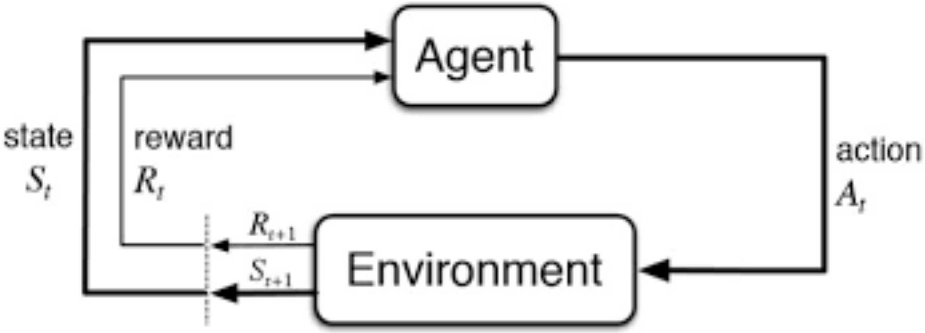
Outcomes:

- **BUST** – player has more than 21 points (dealer does not take cards if all players bust)
- **WIN –** dealer busts or non-busted player has more points than non-busted dealer
- **LOOSE –** non-busted player has fewer points than non-busted dealer
- **PUSH –** non-busted player and non-busted dealer have same points

Players observe cards, make decision, get reward

Reinforcement learning

- Players observe the state of the environment
  - Player’s two cards
  - Dealer’s face up cards
- Strategy: might choose to STAND if dealer has a “bad” card
  - Dealer has a Six of Clubs
  - Player has Four of Hearts and Eight of Diamonds (12 points)
  - Assume dealer’s face down card is a 10
  - If true, dealer must HIT 16, will BUST if next card is 6 or greater
- Training phase
  - Randomly HIT or STAND in each state
  - Track average reward for decisions made over thousands of hands



- Reward:
  - +1 WIN
  - -1 LOOSE or BUST
  - 0 PUSH
- Give reward to each action in a round
  - Player has Five of Diamonds and Six of Clubs cards (11 points) and HITs
  - Gets Three of Hearts (14 points) and HITs again
  - Gets Seven of Spades (21 points) and STANDS
  - Each HIT/STAND decision should get reward

Training: play many hands and track average reward in Q matrix

Q matrix

You'll have to deal with a "soft" Ace also

Average reward if HIT

Player points

Dealer points

	12	13	14	15	16	17	18	19	20
2									
3									
…									
11									

Average reward if STAND

Player points

Dealer points

	12	13	14	15	16	17	18	19	20
2									
3									
…									
11									

Use a three-dimensional array

- Player points
- Dealer points
- Action (HIT or STAND)

See course web page for tips on quickly calculating average

Play phase: always choose the optimal action, useful for end of class tournament

We will have a tournament on the last day of class

- Your player program will connect to my dealer program
- Three tables of six teams
- Top two teams from each table advance to final round
- Ultimate champion will be crowned!
- I've provided a pre-compiled dealer (with debug info) program for testing

Ideas for prize for champion?

Agenda

1. Project details
2. Implementation
3. Grading
4. Tips

Implement a dealer and a player module that communicate over TCP/IP sockets

Dealer program

- Runs from the command line
- Takes number of games to play and port number as parameters (mine also takes number players)
- Sets up a server socket listening for clients (players) to connect
- Once a client connects (you need only handle one client, mine will handle up to six players), pass messages back and forth over socket
  - Create a deck of 52 cards for each game
  - Shuffle the deck
  - Deal cards to the player by sending messages with the card suit and rank as a string (e.g., “Seven of Hearts”)
  - Receive HIT/STAND decisions from the client
  - Calculate the game’s result (WIN, LOOSE, BUST, or PUSH) and send a message to the client
  - Reset and play again (you decide on how many games to play)
- Send a QUIT message to the client when done

Player program should have two modes: training and play

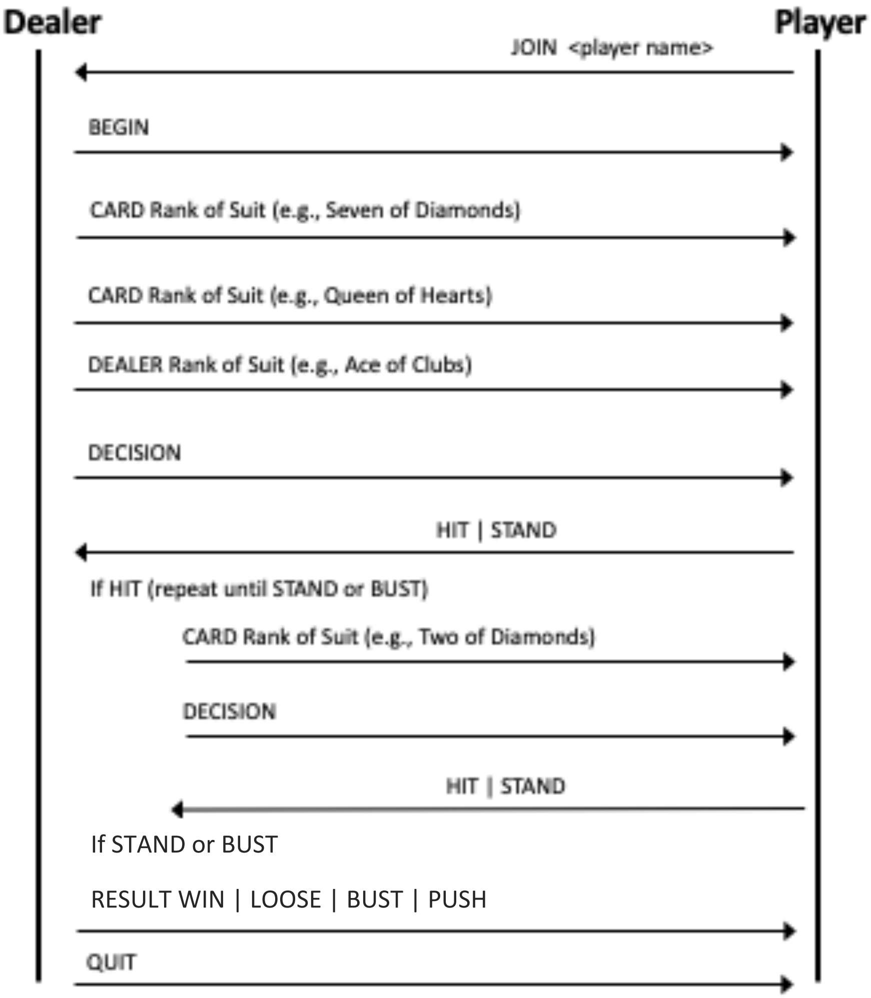
IMPORTANT
PORT = 8080 + team number!

Otherwise, we might have issues with other team's communications

Player program

- Runs from the command line taking the player's name, server's IP address, and PORT number as parameters
- Connects to the server using a socket
- **Training mode** - plays many games with the dealer program
  - Choosing random actions in each state
  - Must be able to write its Q tables to disk and read them back
  - Must be able to continue training after reading the Q table from disk
- **Play mode** - makes optimal decisions based on what it learned during training
  - Reads Q table written to disk during training
  - Uses table to make optimal decisions for each state (e.g., dealer and player cards)

Follow these message passing guidelines



Dealer

Player

JOIN <player name>

BEGIN

CARD Rank of Suit (e.g., Seven of Diamonds)

CARD Rank of Suit (e.g., Queen of Hearts)

DEALER Rank of Suit (e.g., Ace of Clubs)

DECISION

HIT | STAND

If HIT (repeat until STAND or BUST)

CARD Rank of Suit (e.g., Two of Diamonds)

DECISION

HIT | STAND

If STAND or BUST

RESULT WIN | LOOSE | BUST | PUSH

QUIT

Messages

- JOIN <player name>: player asks to join game, player name should not have spaces (use underscore for spaces e.g., team_one_is_here)
- BEGIN: to keep dealer and player in sync, if a player gets a BEGIN message, they should reset for a new game (e.g., discard any cards)
- CARD: dealer sends player a card, rank will be strings “Two” through “Ten”, “Jack”, “Queen”, “King”, or “Ace”, suit will be “Diamonds”, “Hearts”, “Clubs” or “Spades” (e.g., “CARD Nine of Hearts”)
- DEALER: dealer tells player the Rank of Suit of dealer’s face-up card (e.g., “DEALER Ace of Clubs”)
- DECISION: dealer asks the player to make a decision (either HIT or STAND)
- HIT or STAND: player tells the dealer their decision based on player’s cards and dealer’s face-up card (repeat until STAND or BUST)
- RESULT: dealer tells the player if they WIN, LOOSE, BUST, or PUSH (followed by BEGIN if playing multiple rounds)
- QUIT: dealer tells player to quit

Agenda

1. Project details

2. Implementation

3. Grading

4. Tips

Grading

	Points
Peer evaluation	10
Documentation	15
Testing	10
Coding style	10
Functionality	
• Cards/game rules	10
• Dealer module	14
• Player module	12
• Network module	9
Total	45
Makefiles	5
Memory leaks (valgrind)	5
Total	**100**
Extra credit	10

Peer evaluation:

- Survey to get sense of each team member’s contributions
- We can also see GitHub commits!

Extra credit:

- Implement a text-based Graphical User Interface using ncurses
- Up to 10 points available

Agenda

1. Project details
2. Implementation
3. Grading
4. Tips

Tips

Define what module you'll need, consider:

1. Cards module used by dealer and player that models
   - Individual cards
   - A deck of 52 cards
   - A hand of cards, which are the cards a player or dealer holds
2. Network module handles
   - Server set up/tear down
   - Socket connection/close
   - Message passing from dealer to player, and player to dealer
3. Dealer – handles game play, decides on outcome
4. Player – implements reinforcement learning, with train and play modes (use my dealer program to test your player program for the tournament)

Think carefully about how modules will interact

- Write your documentation first so everyone knows what to deliver (think of an interface from CS 10)