Race to a Dollar Game
by Kush Patel

- Title: Race to a Dollar Game
- Team Member Names: Kush Patel
- Problem Statement
  -Many preschoolers need to learn how to count coins, what coins equal a specific amount of money, or just in case you don't have any cash, you can learn from this game by earning which coins equal a specific amount
  -If you gave a person or paid someone in cash over the required amount, then it can help you decide how many coins or amount you have left over
  -It helps kids understand money and how it works through this game
  -The game is mostly for preschoolers, autistic people, little kids, and people who have just moved to the United States who don't know what the United States currency is
  -Two players take turns playing the game, the player that goes first is the human player, in which the human must give an input by choosing what coin to add to their collection to reach a target amount, there will also be an AI player that will compete against you
  -It teaches the kids what coins to add to reach an exact target amount, like what coins and how many of each type of coin equal $0.49
  -A player can only pick valid coins that would result in a value that is less than or equal to in the target, each player can choose from a fixed set of coins, the penny(0.01),nickel(0.05),dime(0.10), quarter(0.25), the first to reach the exact target amount wins the game
  -It enforces a rule in which the player can not exceed the target amount
  -It keeps track of the current amount of coins each player has
  -It displays the game state, current player, valid moves, and a winning message once the game ends.

- Approach
  -I used the alpha-beta pruning algorithm provided in the textbook for the programming project 2 to implement the AI player, but I made a slight modification by adding a depth limit so it prevents doing an infinite search and chooses the best option, if it isn't unsure of which option to pick given that they are all equally good then the depth tracker stops its search and forces it to pick and option
  -Bigger coin values are more prioritised in the search due to it being the best option, but its not guaranteed that they pick the best option, which is the reason why the alphabeta pruning is there and the depth tracker prevents the infinite search
  -The RaceToDollarGame class inherits from the Game class provided in class for the Programming Project 2, and I slightly modified the game state to include the coin list of each player and the allowed moves for each player at a given state.
  -The possible actions are the valid coin values that each player can add to their target amount, and they can't exceed the target amount
  -The game state tracks whose turn it is, the utility, board values, coin histories, current amount and valid moves.
  -There is a possible action list which lists all the possible actions at the beginning of the game

-There is a result function that shows the state of the game after a player takes a move
-A Utility function returns +1 for player 1 win, -1 for player 2 win, and 0 for in-progress.
-There is a terminal test function which checks to see if a player reaches a target amount or if there are no moves left
-The game can be played on a terminal or a GUI but you have to modify it and change the target amount manually

- Description of Software
  -The game consists of a GUI which was written using the Tkinter library, the Tkinter library puts the game in as its parameters to keep the game from running and continuing
  -It was written all in the Python Programming Language
  -The AI Algorithm it uses is Alpha-Beta Pruning with a slight modification, which is a depth tracker which prevents the AI from searching infinitely
  -The RaceToDollar game class was a class that was inherited from the Game Class provided from Programming Project 2
  -The actions(state) function returns a list of valid actions
  -The result(state, move) function results in the new state after a move.
  -The GUI is built using the Tkinter library, it displays the game state visually, it lets the human click buttons to take a move, and the AI player moves automatically after every 5 seconds
  -The state and display of the game GUI change based on the result method. It calls the result method to change the GUI
  -The Game GUI is also a turn-based interface
  -The game starts with Player 1 (AI) taking a move after 5 seconds.
  -Buttons for each coin value are only visible when it's the human player's turn.
  -Real-time Updates of game state after each move.
  -Terminal messages are printed to help debug the game's internal logic

- Evaluation
  -We decided to let the Human player make the first move instead of the AI player because if the AI player always went first, then they would always win
  -There are some situations in which the human player wins, and there are some situations in which the AI player wins, but the AI player always makes the best decision by determining which move it should pick and if it is still searching
  -AI usually makes the best move whenever it is playing a game with a human player, and because of this we have to let the human player go first

- Conclusion
  -I learned that if there is a game in which the AI has trouble making a decision, given that all the moves are equally good, then we must add a depth tracker to prevent the AI algorithm from infinitely searching
  -I learned that this game isn’t for not only for preschoolers who need to know basic knowledge about money, but also for all other people who don’t have any dollar bills but just loose coins in their pockets so they can use them as an emergency just in case

-An Idea for improvement is that we can include it on a Web web-based front End which is one thing I can improve on
-I can add styling to my GUI, like colouring and proper formatting to the GUI
-I can also add a reset to the game to make the game start all over again
-I found out why my program was running in an infinite loop because the AI was continuing searching, and because of that I decided to add a depth limit tracker
-I learned that I can't call the play_game function in the tkinter library because it would play a game inside the GUI without letting the human pick an option so I decided
-I wish I had modularised the structure of the project by splitting the game GUI into a separate file
-This is a turn-based game where each player takes turns when playing the game. There is an human player, and there is an AI player

- References and Sources
  -The sources I used in this class were from the code samples Dr.Pangadan provided to us in class which was the game.py, util.py, and the alpha beta pruning algorithm with a slight modification
  -Here is the source code I used for my AI before I added an modification:
  https://github.com/aimacode/aima-python
  -Here is the Tkinter tutorial I used to learn how to use Tkinter:
  https://www.geeksforgeeks.org/python-tkinter-tutorial/?ref=outindfooter