Tower Blaster

Description:

Tower Blaster is a game that involves re-arranging a group of bricks in order to have an increasing sequence. The user's moves are decided by the user playing the game, by asking for input, and the computer's moves are decided by the program.

One of the important parts during the implementation of this game is to come up with a reasonable enough strategy (algorithm) that ensures a human user does not always beat the computer.

There is an online version of Tower Blaster. Go here:
https://www.gamefools.com/onlinegames/free/towerblaster.html

Try to get a feel for the game. Our job will be similar, but read our specifications. Most importantly, we won't have levels. Playing the game while developing your code can be helpful. A Tower Blaster game starts with a main pile of 60 bricks, each numbered from1 to 60. Think of the numbers on the bricks as the width of the bricks. The objective is to be the first player to arrange 10 bricks in your own tower from lowest to highest (from the top down), because the tower will be unstable otherwise. The bricks in the main pile are shuffled at the start and both the user and the computer are dealt 10 bricks from the main pile. As a player receives each brick, they must place it on top of their current tower in the order it is received. Initially your tower is likely to be unstable.

After the first 10 bricks are dealt to the user and the computer, there will be 40 bricks remaining in the main pile. The top brick of the main pile is turned over to begin the discarded brick pile. On each player's turn, the player chooses to either pick up the top brick from the discard pile or to pick up the top brick from the main pile. The top brick from the discard pile is known. In other words, the discard pile is 'face up' and everyone knows how wide the top brick is. The main pile is 'face down'. Choosing the top brick from the main pile can be risky, because the player does not know what the brick is. Once a player chooses a brick, either from the discard pile or from the main pile, the player decides where in the tower to put the brick. The tower is always 10 bricks high, so placing a brick means that an existing brick in the tower is removed and replaced with the new brick.

If the player takes a brick from the main pile (the one that is 'face down'), the player can reject it and place it in the discard pile. This means that nothing in that

player's tower changes during that turn. If the player takes a brick from the discard pile (the one that is 'face up'), the player MUST place it into the tower.

The first player to get their 10 bricks in order wins.

If, at any point, all of the cards have been removed from the main pile of bricks, then all of the cards in the discard pile are shuffled and moved to the main pile. Then the top card is turned over to start the new discard pile.

Below you will find the explanations of some functions that you might need, but not enough for this whole program. Implement the functions you deem as necessary and create your own functions to make this game work.

Note that in both the main pile and the discard pile, you should only have access to the top.

Functions for reference:

setup_bricks():

- Creates a main pile of 60 bricks, represented as a list containing the integers 1 –60.
- Creates a discard pile of 0 bricks, represented as an empty list.
- This function returns both lists.

shuffle_bricks(bricks):

- Shuffle the given bricks
- This function does not return anything.

check_bricks(main_pile, discard):

- Check if there are any cards left in the given main pile of bricks.
- If not, shuffle the discard pile and move those bricks to the main pile.
- Then turn over the top card to be the start of the new discard pile.

check_tower_blaster(tower):

•Given a tower, determine if stability has been achieved.

•Remember, stability means that the bricks are in ascending order.

•This function returns a boolean value.

get_top_brick(brick_pile):

•Remove and return the top brick from any given pile of bricks. This can be the main_pile, the discard_pile, or your tower or the computer's tower.

•It is used at the start of game play for dealing bricks. This function will also be used during each player's turn to take the top brick from either the discarded brick pile or from the main pile.

•Note: Brick piles are vertically oriented structures, with the top having index 0.

•This function must return an integer.

deal_initial_bricks(main_pile):

•Start the game by dealing two sets of 10 bricks each, from the given main_pile.

•Make sure that you follow the normal conventions of dealing. So, you have to deal one brick to the computer, one to the user, one to the computer, one to the user, and so on.

•The computer is always the first person that gets dealt to and always plays first.

•Remember that the rules dictate that you have to place your bricks one on top of the other. In the earlier picture, this would mean that someone was dealt 46, 33, 10, ..., in that order.

•This function returns a tuple containing two lists -one representing the user's hand and the other representing the computer's hand.

add_brick_to_discard(brick, discard):

•Add the given brick (represented as an integer) to the top of the given discard pile.

•This function does not return anything.

find_and_replace(new_brick, brick_to_be_replaced, tower, discard):

•Find the given brick to be replaced (represented by an integer) in the given tower and replace it with the given new brick.

•Check and make sure that the given brick to be replaced is truly a brick in the given tower.

•The given brick to be replaced then gets put on top of the given discard pile.
•Return True if the given brick is replaced, otherwise returnFalse.

Computer_play(tower, main_pile, discard):

•This function is where you can write the computer's strategy (algorithm).

•Note: Given a particular tower and a particular brick (either from the discarded brick pile or as the top brick of the main pile), you either always take the brick or always reject it.

•Here are some potential decisions the computer has to make:

1. Given the computer's current tower, do you want to take the top brick from the discarded brick pile or do you want to take the top brick from the main pile and see if that brick is useful to you.

2. How do you evaluate the usefulness of a brick and which position it should go into.

3. There might be some simple rules you can give the computer. For instance, it is disastrous (provably so) to put something like a 5 at the very bottom of the tower. You want big numbers over there.

•You are allowed to do pretty much anything in this function except make a random decision or make a decision that is obviously incorrect. For instance, making your bottom brick a 5 is not very smart and a recipe for disaster.

•Also, the computer CANNOT CHEAT. What does that mean? The computer cannot peek at the top brick of the main pile and then make a

decision to go to the discard pile. Its decisions should be something that a human could be able to make as well.

•This function returns the new tower for the computer.

main():

•The function that puts it all together.

•You play until either the user or the computer gets Tower Blaster, which means their tower stability has been achieved.

•Assemble the functionality in the main function properly.

•All user input should take place in the main function.

User Interface:

You're free to create your own user interface for the game, as long as it makes sense for the user playing.

The computer goes first. The program can print the computer's initial tower (in list form) at the beginning of the game, but for the rest of the game, the human player shouldn't see what's in the computer's tower. After the computer's turn, the program should print the results of the turn.

When it's the human user's turn, the program should print the user's tower (in list form), the top brick of the discard pile, and, if they choose to pick from the main pile, the top brick of the main pile.