Project 3

Fair Division, with Linked Lists

Introduction

This project will solve the fair division problem using the method of sealed bids, just as in Project 2. However, in this project, we will use linked list instead of arrays throughout the program and related classes.

Dividing the Assets

The items to be divided will be called the *assets* and the people among whom they are divided will be called the *players*. We will assume that we have m assets and n players. Any positive values are possible. That is, we could have any of the possibilities $m < n$, $m = n$, or $m > n$.

The procedure begins with each player placing a private *bid* on each asset. Their bids represent the values that they place on those assets. Once all the players have placed their bids, the bids are opened and compared. For each asset, the player who placed the highest bid on that asset receives that asset.

The Cash Settlement

The players who received assets add up the values of those assets, based on their bids. If the value of the assets is more than their fair share, then they pay the difference into the *kitty*. If it is less than their fair share, then they receive the difference from the kitty.

At this point, all the players have received exactly their own fair share. However, unless all the bids for each asset were equal, there is money left over in the kitty. This remaining amount is divided evenly among all the players.

The `Player` Class

To facilitate the design of this program, we will create a `Player` class. This class is described in detail in the document `The Player Class`. A `Player` object contains the player's name

and a linked list of bids.

The `Strng` Class

The players' names will be stored in the `Player` objects as strings using the `Strng` class, which was introduced in Lab 4. This class has two advantages over the standard `string` class. First, by using quotation marks as delimiters, we can embed white space in a string. Thus, a player may have a name such as `"John Doe"` or `"Billy Bob 'Sassafras' Calhoun"`. Secondly, if we used the standard `string` class, then it would be tricky to read a linked list of strings because the commas used as separators between the names would be read as part of the names. With the quotation marks as delimiters in the `Strng` class, the commas will be outside the strings and unambiguously not part of the strings.

The Program

The application program should define and use the following functions.

- `LinkedList<Strng> readAssets()` – Reads the names of the assets from a file into a linked list and returns the linked list.
- `LinkedList<Player> readPlayers()` – Reads the players as `Player` objects from a file into a linked list and returns the linked list.
- `LinkedList<double> getFairShares(LinkedList<Player>& player)` – Returns a linked list of the players' fair shares.
- `LinkedList<int> assetSettlement(LinkedList<Player>& player)` – Returns a linked list of indexes (`int`s), one index for each asset. Each index refers to the player who was awarded the asset.
- `LinkedList<double> cashSettlement(LinkedList<Player>& player, LinkedList<int>& recipient, LinkedList<double>& fairShare)` – Returns a linked list of `double`s that represent the cash settlement for the players, one value for each player. A positive quantity indicates that the player *paid* that amount into the kitty. A negative quantity indicates that the player *received* that amount from the kitty. The parameter `recipient` is the list of indexes of players who won the assets, the same list that was returned by `assetSettlement()`.
- `void reportSettlement(LinkedList<Player>& player, LinkedList<double>& fairShare, LinkedList<Strng>& asset, LinkedList<int>& recipient, LinkedList<double>& cash)` – Outputs a report showing the players' fair shares, the settlement of assets (who received what), and the cash settlement (who paid and who received how much).

Place the files `FairDivision.cpp`, `player.h`, `player.cpp`, and `linkedlist.h` in a folder named `Project_3` and place it in the drop box. Your work is due by Wednesday, March 8.