Class design

- Naming conventions: https://www.geeksforgeeks.org/naming-convention-in-c/

Position

Represents a 2D grid coordinate.

Public Data Members

- `int x, y`: Horizontal and vertical coordinates.

Public Member Functions

- `int distanceTo(const Position& other) const`: Manhattan distance to `other`.
- `bool isAdjacentTo(const Position& other) const`: `true` if `other` is in any of the 8 neighboring cells.
- `string str() const`: Serialize the grid as string format `"x y"` (e.g. `"3 5"`).

Items

Represents a collection of item strings (e.g., ingredients or dishes).

Public Member Functions

- `Items()`: Construct an empty `Items` object.
- `Items(const string& rawString)`: Parse a string like `"DISH-ICE_CREAM"` and store each token as an item.
- `void setItems(const string& rawString)`: Update the current item list using a dash-separated string.
- `bool hasItem(const string& item) const`: Check whether the item exists in the current collection.
- `bool hasAllItems(const Items& other) const`: Check whether all items in `other` are contained in the current object.
- `bool isEmpty() const`: Return `true` if no items are held.
- `const vector<string>& getItems() const`: Get a read-only reference to the current list of items.

Private Data Members

- `vector<string> items_`: Stores the list of items.

Customer

Represents a customer’s order and the reward given upon fulfilling it.

Public Member Functions

- `Customer()`: Default constructor.
- `Customer(const string& itemStr, int awardValue)`: Construct with desired items and reward.
- `void setItems(const string& itemStr)`: Update the required items.
- `void setAward(int awardValue)`: Update the reward value.
- `const Items& getItems() const`: Get the required items for the order.
- `const int& getAward() const`: Get the reward amount.

Private Data Members

- `Items item_`: Required items for fulfilling the order.
- `int award_`: Points awarded after fulfilling the order.

Table

Represents a table with a position and the items on it.

Public Member Functions

- `Table()`: Default constructor.
- `Table(const Position& pos, const string& itemStr)`: Construct a table at the given position.
- `void setPosition(const Position& pos)`: Set the position of the table.
- `void setItems(const string& itemStr)`: Update the items on the table.
- `Position getPosition() const`: Retrieve the table's position.
- `Items getItems() const`: Retrieve the items currently on the table.

Private Data Members

- `Position pos_`: The location of the table.
- `Items items_`: Items currently placed on the table.

Chef

Public Member Functions

- `Chef()`: Default constructor; object must be initialized via `update()` before use.
- `Chef(int x, int y, const string& itemStr)`: One-shot initialization of position and inventory. Pass in `itemStr` like `"DISH-ICE_CREAM"` to `item_`
- `void update(int x, int y, const string& itemStr)`: Set or reset the chef's state for this turn. `itemStr`: dash-separated tokens, e.g. `"DISH-APPLE-ICE_CREAM"`
- `bool isEmptyHanded() const`: Returns `true` if the chef holds no items.
- `string doAction(const string& action, const Position& target = {-1,-1}, const string& comment = "") const`
  Build a turn command string:
  - `"WAIT; comment"` if `action=="WAIT"`.
  - Otherwise `"ACTION x y; comment"`, e.g. `"MOVE 3 5; chop dough"`.
- `string dropItem(const Position& dropPos, const string& comment = "drop") const`
  Issue a drop action:
  - If empty-handed, returns `"WAIT; nothing to drop"`.
  - Else returns `"USE x y; comment"` at `dropPos`.
- `Position getPosition() const`: Returns the chef's current `Position`.
- `const vector<string>& getItems() const`: Returns a const reference to the held item tokens.

- `bool hasItem(const string item)`: Returns `true` if the chef's inventory contains that token.
- `bool canServeCustomer(const Customer& customer) const`: Returns `true` if the chef has all items required by `customer.getItems()`.

Private Member Function

- `void ensureInitialized() const`
  Asserts that the chef was initialized by `update()` or the parameterized ctor.

Private Data Members

- `Position pos_`: Current coordinates.
- `Items item_`: Parsed inventory tokens.
- `bool initialized_{false}`: Tracks whether the chef has been initialized.

Kitchen

Public Member Functions

- `void initMap()` : Initialize the kitchen map.
- `void setTableState()` : Update what's on the table.
- `Position getClosestEmptyTable(const int &x, const int &y)` : Find the nearest empty table.
- `vector<Position> getPosition(const string &name)` : Get the positions of specified items or equipment.
- `void printMap()` : Debug print for the kitchen map.
- `void printEquipment()` : Debug print for all equipment locations.
- `void printTable()` : Debug print for all tables and their items.
- `vector<vector<char>> getMap() const` : Getter for the map layout.
- `unordered_map<string, Position> getEquipment() const` : Getter for equipment positions.
- `vector<Table> getTable() const` : Getter for the list of tables.

Private Member Functions

- `bool isInside(int x, int y)` : Check if a coordinate is within map bounds.
- `bool isTableOccupied(const Position& pos)` : Check if a table at this position is occupied.

Private Data Members

- `vector<vector<char>> map_`: The kitchen map grid.
- `unordered_map<string, Position> equipment_`: Stores equipment name and their positions.
- `vector<Table> tables_`: List of all tables and their items.