Explanation of your algorithm:

The goal was to optimize the food and drink choices, and I made the assumption that you will want to provide something, even if not preferred, to all party attendees. To do that, my algorithm will first determine the overall cheapest (lowest costing) food and drink combo that exists within all the possible food and drink items that can be bought for the party. Picking the cheapest food and drink item represents the lowest costing food and drink combo (referred to as GLCC for Global Lowest Costing Combo). The combination of these two represents the cheapest possible food and drink combination that can be given to all attendees.

Then, the algorithm proceeds to determine the cheapest food and drink combination for each party attendee based on their preferences (referred to as ILCPC for Individual Lowest Costing Preferred Combo). It establishes the ILCPC for each attendee by choosing separately sorting a given attendee’s food and drink choices by unit cost in ascending order, then choosing the lowest food and drink item, thus creating the ILCPC for the individual. This list of ILCPC’s for all party attendees is used later in the algorithm.

However, given that this the cheapest food and drink combination, the algorithm checks to see, given the budget, if all the party attendees can be given this lowest food and drink combo. It checks this by determining if the budget is less than the cost of the GLCC \* the number of party attendees. If it is, the algorithm states that this party cannot be held since there is not enough money to give each attendee at least the GLCC. If the budget is enough to at least assign the GLCC to every party attendee, the algorithm continues.

Assuming the aforementioned budget check was passed, the algorithm first sorts all ILCPCs based on the cost of each ILCPC in ascending order. Then it proceeds to determine how much money is remaining if the GLCC was simply assigned to all attendees. This remaining budget is calculated by subtracting the cost of the GLCC \* the number of party attendees from the overall given budget.

This remaining budget is signified as the “rem\_budget” in the algorithm. Using this leftover money, the algorithm loops through the sorted ILCPCs to determine if there are ILCPCs that can be assigned to individuals instead of simply giving them the GLCC, which they may not prefer. The algorithm determines this by adding the already allotted cost of the GLCC and the leftover money, and then seeing if an ILCPC is less than this amount (implying that the algorithm can afford to give an individual one of their preferred food and drink combinations over simply giving them the GLCC). If the algorithm can afford this ILCPC, it assigns the individual their preferred combination and decreases the remaining budget by the difference in cost of the ILCPC and the GLCC (the cost of choosing the preferred combo over the base combo). However, in the case where the algorithm cannot afford to give the ILCPC of an individual, it simply assigns the GLCC to the individual. It does not decrement the remaining budget since it could not afford to decrease the remaining budget. These set of steps are performed for all the ILCPCs, and thus results in assigning all attendees with either their lowest costing preferred food and drink combination or the overall lowest costing food and drink combination.

Test cases:

Global Min = Minimum budget required to give every party attendee the cheapest food and drink combination

Preferred Min = Minimum budget required to give every party attendee their cheapest preferred food and drink combination

1. **Budget < Global Min**

This test case was important to ensure that the algorithm correctly determined that the budget for the party was not enough to even assign each party attendee the cheapest food and drink combination. Given I made the assumption that every party attendee **must** be provided something, even if not preferred, the algorithm correctly identified that the party cannot be hosted and returned the following screen:

ENTER SCREENSHOT HERE.

1. **Budget = Global Min**

This test case was important to ensure that the algorithm correctly provided every attendee the overall cheapest food and drink combination because that is the only set of food and drink combinations that the budget allowed. The output for this can be seen here:

1. **(Budget > Global Min) and (Budget < Preferred Min)**

This test case was particularly important to ensure that the algorithm assigned certain attendees their preferred lowest costing combination and certain attendees the overall lowest costing combination. This can be seen to be true based on the fact that

4) Budget = Preferred Min

5) Budget > Preferred Min

Assumptions:

Person -> food + drink