ASSESSMENT FORM

Imagine you are building a billing system for a cloud storage service.

Each customer has their own tiered pricing, where they get charged based on the storage they have used. You need to calculate the total bill of each customer given their pricing model. We need to solve this for N customer, with each having a different pricing model.

Example:

Customer 1:

1GB-100GB - $1/GB

101GB-1000GB - $0.5/GB

1001GB-10000GB - $0.25/GB

10001GB+ - $0.1/GB

Customer 2:

1GB-50GB - $1/GB

51GB-1000GB - $0.4/GB

1001GB-100000GB - $0.25/GB

100001G+B+ - $0.05/GB

Bank of Amercia:

1GB-100GB - $1/GB

101+ - $0.01/GB

Customer 4:

1GB+ $0.25/GB

Example: Customer 1 uses 150GB of storage their total bill would be = $125 ($1 per GB up to 100GBs, then $0.5/GB for remaining 50GBs)

Example: Customer 2 used 1500GB their total bill would be = $555 ($1/GB up to 50GBs, $0.4/GB for the next 950GBs & then $0.25 for the remaining 500GBs)

Problems to solve.

1. Come up with a data structure that will represent customer pricing tiers.

2. Come up with an algorithm that will use that datastructure & calculate total cost.

3. Return or print total cost.

1. To represent customer pricing tiers, a data structure that could be used is a list of dictionaries, where each dictionary represents a pricing tier for a particular customer. The dictionary would have the keys "min\_storage" and "max\_storage" to represent the range of storage for that pricing tier, and "price" to represent the price per GB for that tier. For example, the pricing tiers for Customer 1 would be represented as:

[

{'min\_storage': 1, 'max\_storage': 100, 'price': 1},

{'min\_storage': 101, 'max\_storage': 1000, 'price': 0.5},

{'min\_storage': 1001, 'max\_storage': 10000, 'price': 0.25},

{'min\_storage': 10001, 'max\_storage': float('inf'), 'price': 0.1}

]

1. An algorithm to calculate the total cost for a customer could be as follows:

* Initialize a variable "total\_cost" to 0
* Iterate through the pricing tiers for the customer
* For each tier, check if the customer's storage usage falls within the range of that tier (min\_storage <= usage <= max\_storage)
* If it does, calculate the cost for that tier by multiplying the usage in that tier (usage - min\_storage + 1) by the price for that tier
* Add the cost for that tier to the "total\_cost" variable
* Repeat for the next tier until all usage has been accounted for

1. To return the total cost, the algorithm from step 2 can be implemented in a function that takes in the customer's storage usage and pricing tiers as arguments, and returns the total cost. Alternatively, the total cost could be printed within the algorithm.