Develop a small HTTP-based back-end service which stores and provides scores and ranks for customers.

1. Business Model

- **Customer** : Each customer is identified by an unique ***CustomerID,*** which is an int64 number. In this service, any arbitrary number can identify a customer.
- **Score**: Each customer has a score, which is a decimal number. The default score of a customer is zero.
- **Leaderboard**: All customers whose score is greater than zero participate in a competition. Each customer is associated with an unique rank in the leaderboard, determined by their scores.
  The with the highest score is at rank 1
  The customer with the second highest score is at rank 2

  ...
- Two customers with the same score, their ranks are determined by their CustomerID, lower is first.
- When customer' s score changes, its rank in leaderboard is affected in realtime.

Here goes an example

Customer ID	Score	Rank
15514665	124	1
81546541	113	2
1745431	100	3
76786448	100	4
254814111	96	5
53274324	95	6
6144320	93	7
8009471	93	8
11028481	93	9
38819	92	10

2. Nonfunctional Requirements

- No persistence nor database is required. State in memory only.
- Only use .Net Core. Do not use external frameworks/assemblies, except maybe for testing.
- The service needs to be able to handle lots of simultaneous requests
- The service needs to be able to handle lots of customers, so bear in mind the complexity, especially the time complexity of frequent operations.
- Create a repository in github.com, and upload your source code there.

3. Functional Requirements

You have to implement the following 3 methods.

3.1 Update Score

```
Request: POST /customer/{customerid}/score/{score}
Response: Current score after update
```

- **{customerid}** : arbitrary positive int64 number
- **{score}** : A decimal number in range of [-1000, +1000]. When it is positive, this number represents the score to be increased by; When it is negative, this number represents the score to be decreased by.

Initially, there is no customer in the leaderboard.

When this method is called and the specific customer does not exist in leaderboard, the customer is added in leaderboard and its score is stored.

If the specific customer already exists, then its score is updated. E.g. If score of customer 3333333 was 80, now update the score with -20, then the score in response is 60. Idempotence is not needed.

3.2 Get customers by rank

`Request: GET /leaderboard?start={start}&end={end}`
`Response: A JSON structure represents the found customers with rank and score.`

- **{start}** : start rank, included in response if exists
- **{end}** : end rank, included in response if exists

E.g. Given that start rank is 41 and end rank is 44, the response could be something like.

Customer ID	Score	Rank
76786448	78	41
254814111	65	42
53274324	64	43
6144320	32	44

Tips: avoid sorting customers in this API.

3.3 Get customers by customerid

`Request: GET /leaderboard/{customerid}?high={high}&low={low}`
`Response: A JSON structure represents the found customer and its nearest neighborhoods.`

- **{customerid}** : customer to lookup in leaderboard
- **{high}** : optional. Default zero. number of neighbors whose rank is higher than the specified customer.
- **{low}** : optional. Default zero. number of neighbors whose rank is lower than the specified customer.

E. g. Given that *customerid* is 7777777, *high* is 2, *low* is 3. the response could be something like.

Customer ID	Score	Rank
7786448	313	89
54814111	301	90
7777777	**298**	**91**
96144320	298	92
16144320	270	93
2000437	239	94

Tips: avoid sorting customers in this API.