CS 245 - Lab 4 Fall, 2021

Lab 4 - Summer Ice Cream

The goal of this assignment is to demonstrate your mastery of sorting. Download your code here.

Background

Monday was a sweltering summer day. Before class starts, professor Brizan wants to buy some ice cream bars on his way to the school.
At the store, there are `n` ice cream bars. You are given an array `costs` of length `n`, where `costs[i]` is the price of the `i th` ice cream bar in USD. Professor Brizan initially has `dollars` dollars to spend, and he wants to buy as many ice cream bars as possible.
Return the **maximum** number of ice cream bars professor Brizan can buy with `dollars` dollars.
Note: professor Brizan can buy the ice cream bars in any order.

Constraints:

- *costs.length == n*
- *1 <= n <= 10^5*
- *1 <= cost[i] <= 10^5*
- *1 <= dollars <= 10^8*

Examples:

```
Input: costs = [1,3,2,4,1], dollars = 7
Output: 4
Explanation: professor Brizan can buy ice cream bars at indices 0,1,2,4 for a total price of 1 + 3 + 2 + 1 = 7.
```

```
Input: costs = [10,6,8,7,7,8], dollars = 5

Output: 0

Explanation: professor Brizan cannot afford any of the ice
cream bars.
```

```
Input: costs = [1,6,3,1,2,5], dollars = 20

Output: 6

Explanation: professor Brizan can buy all the ice cream bars
for a total price of 1 + 6 + 3 + 1 + 2 + 5 = 18.
```

Requirements

- You must submit a single Java file containing the class named **IceCreams**. You class must contain the function named "`maxIceCreams`" with the following header:

```
public int maxIceCreams(int[] costs, int dollars)
```

- You may not use the Arrays.sort() function, you need to implement your own function for sorting. (think about what sorting you would use in terms of time and space efficiency, there's no mandatory requirement to which sorting you need to implement, but it is a good practice for your project 1.)
    - When you are iterating the sorted array, when is a good time to stop looping? (ie. price of the ice cream, dollars you have at the moment)

Submission

- Download the source code for your implementation from Github Classroom. If you don't have a Github account, create one via github.com > Sign up.
- Select your name from the list of names available, which will link your GitHub ID. If you do not see your name on the list, speak with the instructor or one of the teaching assistants.
- Submit the repository link on Canvas. You may also add any comments in a README file (text, PDF or Word document) to help the grader understand or execute your implementation.

Grading

Your grade for this assignment will be determined as follows:

- 75% = Implementation: your class implementations must run successfully with the source files and data provided. It must produce the expected results, a sample of which appears in the Implementation section above. Any deviation from the expected results results in 0 credit for implementation.
- 15% = Decomposition: in the eyes of the grader, the degree to which your solution follows the suggestions above or otherwise represents a reasonable object-oriented and procedural decomposition to this problem.
- 10% = Style: your code must be readable to the point of being self-documenting; in other words, it must have consistent comments describing the purpose of each class and the purpose of each function within a class. Names for variables and functions must be descriptive, and any code which is not straightforward or is in any way difficult to understand must be described with comments.