intelliarts

# Test Problem

## 1. Application overview

Your task is to implement a basic version of a snack vending machine. The solution will manage different snack categories, control the price and amount of available items in each category, handle purchases, and provide related statistics.

**Note:** project might be implemented as a simple *console, web* or *browser-based* application - it’s up to you to decide.

## 2. Requirements

As a result of the test problem solution you should provide an application that accepts and executes the following set of commands:

- ***addCategory*** *“Chocolate bar” 35.75 12* — register a snack category in the system.
  Command accepts the following parameters:
  - *“Chocolate bar”* — the name of the snack category to be served by the vending machine
  - *35.75* — the price of items in the category
  - *12 (optional, default 0)* — the number of purchasable snack items in the category

- ***addItem*** *“Chocolate bar” 25* — register provided amount of snack items to sell.
  Command accepts the following parameters:
  - *“Chocolate bar”* — the name of the snack category added items belong to
  - *25* — the number of added snack items for sell

- ***purchase*** *“Chocolate bar” 2021-04-13* — purchase a single snack item.
  Command accepts the following parameters:
  - *“Chocolate bar”* — the name of the snack category an item is sold
  - *2021-04-13* — the purchase date

- ***list*** — show list of served categories with amount of items available for sale sorted by amount

- ***clear*** — stop serving all snack categories that don’t have items for sale (items can not be purchased)

- ***report** 2021-04* — show earnings by category in specified month
  Command accepts the following parameters:
  - *2021-04* — the month for which the report is requested

- ***report** 2021-04-21* — show earnings by category gained since provided date till now sorted by category name.
  Command accepts the following parameters:
  - *2021-04-21* — the start date of the period for which the report is requested

## 3. Optional Requirements

In order to get extra points for the test problem solution you might cover your application’s source code with unit tests.

## 4. Application usage example

Here is an example of application usage flow with an output that is shown after execution of each command:

```
> addCategory “Chocolate bar” 32.75 12
```

*Chocolate bar 32.75 12*

```
> addCategory “Donut” 29.5
```

*Donut 29.50 0*

```
> addCategory “Cracker” 18 2
```

```
    Cracker 18.00 2

> purchase “Cracker” 2021-04-21

    2021-04-21
    Cracker 18.00

> addItem “Donut” 3

    Donut 29.50 3

> purchase “Chocolate bar” 2021-04-22

    2021-04-22
    Chocolate bar 32.75

> purchase “Cracker” 2021-04-24

    2021-04-22
    Cracker 18.00

> purchase “Donut” 2021-04-25

    2021-04-25
    Donut 29.50

> list

    Chocolate bar 32.75 11
    Donut 29.50 2
    Cracker 18.00 0

> clear

    Cracker 18.00

> purchase “Chocolate bar” 2021-04-28
```

*2021-04-28*
*Chocolate bar 32.75*

```
> addItem “Donut” 10
```

*Donut 29.50 12*

```
> list
```

*Donut 29.50 12*
*Chocolate bar 32.75 10*

```
> report 2021-04
```

*Chocolate bar 65.50 2*
*Cracker 36.00 2*
*Donut 29.50 1*
*>Total 131.00*

```
> report 2021-04-25
```

*Chocolate bar 32.75 1*
*Donut 29.50 1*
*>Total 62.25*

## 5. Deliverables

Application should be delivered via link to a Github repository. Repository should either

*contain a “dist” folder with pre-built application executables and a README.md file with instructions on how to run an application*

or

*contain README.md file with instructions on how to build and run an application.*

## 6. Timeline

We expect you to implement and provide deliverables in up to 7 days starting from the moment you've received the test problem.

## 7. Evaluation criteria

Test problem evaluation criteria:

- How well the application meets the requirements
- How well the application handles corner cases
- How well an application is implemented from the perspective of OOP/OOD principles or principles of functional programming
- Code readability and maintainability
- Have you managed to cover the code with unit tests

**We wish you good luck and are waiting to meet you at the interview to discuss the provided solution!**