Programming – Final Exam June 20th, 2022	uc3m
Bachelor's degree in Computer Science and Engineering	

READ **CAREFULLY** THESE INSTRUCTIONS BEFORE STARTING THE EXAM, NOT FOLLOWING THEM WILL BE PENALIZED WITH AT LEAST 1 POINT:

- The duration of this exam is **2.5 hours.**
- **Read the entire exam** before writing any code.
- If you have any **doubt,** ask it during the **first 30 minutes**. After that take your own decisions and justify them in the exam. No questions will be answered after 30 minutes.
- Do not answer on this sheet, use the provided ones. This is your personal copy, don´t deliver it, keep it after the exam. Use it as you want.
- Fill in all the pages with a **pen**. Do **not use a pencil or a red pen**.
- Do not forget to include your name and surname on every page.
- Notes, slides, books, etc. are allowed. Electronic devices are not allowed.
- Use a **sheet for each class** and inside it put the methods in **the order** they appear below. Deliver the classes in the order they are introduced below.
- Among others you will be **marked with 0 points** in a section in case of:
  - Using functions, language features or libraries not seen during the course.
  - Using `break` in a loop or in a function.
  - Using any list method; the only exceptions are `append`, `insert` and `remove` (the `del()` and `len()` functions can also be used)
  - Using `sum()`, `max()` or `min()` functions.
  - Using a non-declared variable, a variable belonging to another class or method, or a global variable.
  - In methods for which a header is provided, changing it in any way, not using some of the parameters or altering them.
- **No new attributes** can be created other than the ones required in each section.

Problem 1 (10 Points).- This problem will simulate the calculation of the electricity price in Spain, according to the following concepts:

- Every hour there is a given electrical demand, a positive float number.
- The hourly electrical demand is covered by a series of electricity generators. Each generator offers a given quantity of electricity for each hour at a fixed price, which is common for all hours.
- Generation can be of three different types: gas, renewable or nuclear.
- To fulfill the demand, every hour the generator offering power at the lowest price is selected. More generators are selected in ascending order of price until the accumulated quantity they offer equals the hourly demand.
- All the generators are paid at the price of the last selected one. As an example, let's suppose that for a given hour the power demand is 100, the first generator offers 40 at 10 Euros, the second one 30 at 15 Euros, the third one 30 at 20 Euros and the fourth generator 50 at 25 Euros. As the offers of the 1st, 2nd and 3rd cover the demand and the last selected one charges 20 Euros, the first one will receive 40 x 20 = 800 Euros, the second one 30 x 20 = 600 Euros and the third one 30 x 20 = 600 Euros. The fourth generator will not be paid.

Taking into account the former information, create the following classes and methods. For the methods it is compulsory to include a comment with the **class they belong to**. No new attributes can be created. You can create extra methods if you want.

1. (**1 point**) Class `Generator` with attributes: `name` (str), `kind` (str) that can only be "gas", "renewable" or "nuclear", `offer`, a list of int numbers with the quantity of energy the generator offers every hour,

price (float), the price that generator offers its energy today, and profit (float), the money won by that generator today. The offer will be initially an empty list, the price and the profit will be initially 0. Create an init method receiving the appropriate parameters and a property and a setter for the kind attribute.

2. (**0.5 points**) Class Interval, with attributes hour (int), demand (int) and producers, a list of Generator objects, initially empty. Create an init method with the appropriate parameters. Do not create properties nor setters.
3. (**1 point**) Class Daily_Market with an attribute hours, a list of Interval objects, initially empty, and another attribute generators, a list of Generator objects. Create a property and a setter to check that the generators list is not empty and that all their elements are of Generator type.
4. (**1 point**) A create_offer(self, gas_price: int) method that fills the offer list of a generator object and sets the price it will sell it. Offers will be 24 random numbers in the range 10 to 100. If the generator is renewable or nuclear, the price will be a random number in the range 0 to 30, if it is a gas one, the price will be also a random float in the range gas_price to gas_price * 2.
5. (**0.5 points**) A create_intervals(self) method that fills the hours attribute of the Daily_Market class with a list of 24 Interval objects with random demands in the range 100 to 200.
6. (**1 point**) A sort_generators (self) method that shorts the generators list in ascending order of price using the bubble sort algorithm.
7. (**2.5 points**) A select_generators(self) method that for each Interval looks for the generators that can fulfill its hourly demand, selecting the cheapest ones first. It must append the generators to that interval producers list and update the profit of the chosen ones. Assume there will always be enough generators to fulfill the demand and that the last selected one sells all its offer even in the case it exceeds the remaining demand.
8. (**0.5 points**) A __str__(self)->str method that for each interval returns the demand and the list of producers as in the example below.
9. (**0.5 points**) A __str__(self)->str method that returns the previous information for all the intervals of a day.
10. (**1.5 points**) Implement a main program creating 10 random kind generators with names generator0, generator1, etc. It will create also their offers. Next, it will create a Daily Market with those generators, the intervals and their demands. It will select generators and print the generators selected for each hour and the final profit as in the example.

Execution example (it only shows the first 2 hours, the profit of every generator must be printed below it, here two columns have been used for the shake of brevity)

```
Hour: 0 Demand: 125
generator8 Offer: 15 Price: 1.45
generator6 Offer: 49 Price: 4.24
generator0 Offer: 51 Price: 4.63
generator5 Offer: 38 Price: 11.82

Hour: 1 Demand: 130
generator8 Offer: 75 Price: 1.45
generator6 Offer: 76 Price: 4.24
```

```
Total profit:
generator8: 8862.28
generator6: 9040.91
generator0: 7969.0
generator5: 7092.85
generator4: 1021.79
generator2: 0
generator9: 0
generator1: 0
generator3: 0
generator7: 0
```