Programming – Final Exam January 26th, 2021	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:

- **Read the entire exam** before writing any code.
- **Questions will be answered** only during the **first 30 minutes**. After that take your own decisions and justify them in the exam.
- Do not answer on this sheet, use the provided ones.
- This is your personal copy, keep it after the exam. Use it as you want.
- Fill in all the pages with a **pen** (personal data and answers). 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.
- Problems can be solved in any order, but if so, use a new sheet for each problem and deliver them in the **right order**. Also, deliver the problem sections in order.
- The duration of this exam is **3 hours.**
- 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, except for: append, insert and remove (the del function can also be used)
  - Using a non-declared variable or a variable belonging to another class or method.
  - Using global variables.
  - In methods for which a header is provided, not using any of the parameters or altering its type.
  - Changing the header of any of the provided methods.
- **No new attributes** can be created other than the ones required in each section.

Problem 1 (10 Points).- The 2021 snowstorm in Madrid was the largest in a century and showed that the city is not ready for this kind of rare events. The aim of this exercise is to help the city major to deal with this situation if it happens again in the future by having an automatic system that cleans the streets using snowplows. The main concepts of our problem are the following:

- A `City` has `Streets` and `Snowplows` that can clean `Snowflakes`.
- `Streets` can be horizontal or vertical, they have a starting coordinate (x,y), a length, and a temperature.
- A `Snowplow` is the machine used to clean the `Snowflakes` from the `Streets`, they are at a given coordinate (x,y) and have a certain amount of fuel.
- `Snowflakes` fall at a given coordinate (x,y) belonging to a street, if the street's temperature is below 2 degrees Celsius they will stay, if not, they will melt.

Considering the former information, create a program as follows:

a) (1 point) Create the `Snowflake` class with attributes `x`, `y` and `temperature` (`temperature` must be private). Create an `init` method with the parameters you consider appropriate. All the attributes must be declared inside it and only inside it. Create properties and setters to check that x >= 0 and y >= 0. Create also a read-only property, `melted`, that returns if the snowflake is melted or not.

b) (1 point) Create the `Street` class with attributes `horizontal` or vertical, `starting_x`, `starting_y`, `length`, `temperature` and a list of `snowflakes`, which initially will be the empty list. No properties are needed in this class. Create an `init` method with the parameters you consider appropriate. All the attributes must be declared inside it and only inside it. Include a `str` method that shows the following information (for a vertical street the H-`<starting_y>` must be changed by an V-`<starting_x>`): "`Street H-<starting_y>, goes from (<starting_x>, <starting_y>) to (<end_x>, <end_y>). Temperature: <temperature>`", where <item> must be replaced by its value. See an example below.

c) (0.5 points) Create the `Snowplow` class with attributes: `x`, `y` and `fuel`. As in the previous sections declare all attributes into the init method. No properties are needed.

d) (0.5 points) Create the `City` class. It has two attributes for its dimensions, `height` and `width`, which must be integer numbers bigger than 5 (use properties and setters for them). It also has a tuple of `street` objects and another of `snowplow` objects, both initialized to None. As in the previous sections declare all attributes into the init method.

e) (1 point) Implement the method `create_streets(self)` that creates the streets of a city and fills the tuple with them. It must create `height` horizontal streets and `width` vertical ones. The first horizontal street will be in position (0,0), the second in (0,1), etc. Meanwhile, positions for vertical streets will be (0,0), (1,0), etc. Their `length` will be a random number in the range [5, width] for the horizontal streets and [5, height] for the vertical ones. Their `temperature` will be a random value in the range [-5, 4]. Specify the class this method belongs to.

f) (0.5 points) Implement the method `create_snowplows(self)` that creates the tuple of snowplows of the city. It must create a snowplow placed at the beginning of each street. Their initial `fuel` must be a value in the range [1, 20]. Specify the class this method belongs to.

g) (1.5 points) Implement the method `snow(self, number: int)` that creates `number` snowflakes randomly placed in the streets of a city: their `temperature` must be the one of the street they are placed, their `x` and `y` must be in the limits of the street. Specify the class this method belongs to.

h) (1 point) Implement the method `needs_cleaning(self)-> bool` that specifies if a street needs to be cleaned. A street needs to be cleaned if it has at least two non-melted snowflakes in any of its positions. Specify the class this method belongs to.

i) (0.5 points) Implement the method `streets_to_clean(self)-> str` that returns a string with the streets that need to be cleaned in a city (see example of the output below). Specify the class this method belongs to.

j) (2 points) Implement the method `clean_street(self)` that cleans all the streets needing it. Each snowplow will walk through the street one position each time, removing from the list of snowflakes of the street those ones at the current position. Each time it moves its fuel will be decreased by 1 unit. If it runs out of fuel it will stop cleaning. Specify the class this method belongs to.

k) (0.5 points) Create a main program with a 20x20 city and 100 snowflakes. It must attempt to clean all the streets and print which ones were not cleaned (because the snowplow did not have enough fuel). See an example of the required output below.

Example of execution:

```
Streets to clean:
Street H-15, it goes from (0,15) to (15,15). Temperature: -1
Street V-2, it goes from (2,0) to (2,17). Temperature: -3
Street V-3, it goes from (3,0) to (3,5). Temperature: -1
Street V-5, it goes from (5,0) to (5,16). Temperature: -4
Street V-11, it goes from (11,0) to (11,12). Temperature: -4

Streets not cleaned:
Street H-15, it goes from (0,15) to (15,15). Temperature: -1
```