Data Structures and algorithms Bachelor in Computer Science Universidad Carlos III de Madrid Lab case. COURSE 2019-2020	uc3m | Universidad Carlos III de Madrid

Application to manage products deliveries at Amazon.

This lab case includes three phases. This document describes the first one. The statement for phases 2 and 3 will be published in the following weeks.

Phase 1 – Linear Structures

Amazon identifies packages using the following data:

- A package number (for example, '*132-1352234-332344'*). This identifier must have the following format XXX-XXXXXX-XXXXXX, where X can be any digit.
- Delivery address, which is formed by the name of the street and the corresponding number (for example, '*Federica Montseny street nº7'*), ignoring floor number to make it simpler, and
- Postal Code (for example, *28911*).

Amazon relies on its delivery staff. For each delivery staff member the relevant data is:

- An identifier for each delivery staff member (for example, '*R1000335'*). This identifier must have the following format: RXXXXXX where X can be any digit.
- Name and surname (for example, '*Isabel Segura-Bedmar*').
- A status ('*active*' or '*inactive*'). The 'inactive' status is used to identify those delivery staff members that are absent for different reasons.

Besides, each delivery staff member has assigned one or more delivery zones. Each delivery zone is identified by a postal code. As an example, delivery staff member '*R1000335'*, can be assigned to the following delivery zones: 28911, 28912, 28913, 28914. A zone can be assigned to several delivery staff members.

Everyday, Amazon has to process all the packages that must be delivered that given day (dates can be ignored to make the problem easier). As previously mentioned, each package must be allocated to one of the delivery staff members covering the delivery zone for that package. If, for some reason, the delivery zone has not been assigned to any delivery staff member, that delivery zone will be assigned to the delivery staff member with the lower number of assigned zones. So, when the labour day starts each delivery staff member has allocated a set of packages.

Every delivery staff member must deliver his/her packages in the same order than those packages were assigned. When delivering a package, two situations are possible: i) The package is successfully delivered or, ii) the package could not be delivered. In case i) the data for the delivered package must be stored in a data structure containing data of all packages successfully delivered. For each package successfully delivered is enough to store the package id and the delivery staff member id. Besides, the given package must be removed from the set of packages assigned to the corresponding delivery staff member.

If, for any reason, the package could not be delivered, it must be reassigned as the last package of the set of packages assigned to the given delivery staff member (that is it must be redelivered after the rest of packages assigned to that delivery staff member have been delivered). After the third delivery attempt, the package must be left appart and its data must be stored as an incident. In this situation, only package id, delivery staff member id and the reason of the incident, 'number of delivery attempts exceeded', must be stored.

If a member of the delivery staff must quit delivering her/his packages, each of the pending packages must be reassigned to an active delivery staff member who covers the delivery zone corresponding to the given package. If no delivery staff member is available, the package must be stored as an incident including 'delivery staff member not available' as the reason of the incident.

Please find below the tasks that you must develop as part of this lab case:

- Implement the data structures needed to represent:
    - The information for a given package. The name of the structure musst be **Package**.
    - A data structure containing all the information for the packages to be delivered by the company. The name of this data structure must be **Orders**.
    - The information for the Delivery Staff Members. Please, remember that a Delivery Staff Member has one or more assigned zones and a set of packages to deliver. The name of this structure must be **DSMember**.
    - A data structure storing all the DSMembers at Amazon. The name must be **DSMembers**.
    - Packages successfully delivered. This data structure must be called **Delivered**.
    - Packages that could not be delivered. The corresponding structure must be called **Incidents**.

Besides, you will have to implement the main class, **AmazonManagement**, that simulates the functionality that is described in the following paragraphs. This class must include attributes representing orders, delivery staff members, the packages that have been properly delivered and the packages that have suffered some incident. Obviously, there could be additional attributes. This class must include the following functionality:

- Implement a function, **loadOrders**(), that receives the necessary data to initialize the object that stores the orders (packages) that Amazon must deliver.
- Implement a function, **loadDSMembers**(),that receives the necessary data to initialize the object that stores the distributors of Amazon. In this function, the distributors will not have any order (package) assigned.
- Implement a function, **showDSMembers**(), that shows the list of distributors (identifier, status, zones and assigned packages) in ascending alphabetical order (last name).
- Implement a function, **assignDistribution**() that assigns each order (package) to a distributor. Remember that the distributor must be active and

must cover the area of the package. If there is no active distributor for that zone, the package (and its zone) will be assigned to the active distributor with fewer zones.

- Implement a function, **deliverPackages**(), which simulates the delivery of the packages for a specific distributor. The function receives a distributor (identifier) and must process all its packages, always starting with the first package and processing them in strict order. For each package to be delivered, a random value (True or False) is generated to indicate whether the package has been correctly delivered or not. If the value generated is True, the package must be removed from the set of packages assigned to that distributor, and registered as delivered. If, on the contrary, the value generated is False, which means that the package could not be delivered, the package will be placed at the end of the orders to be delivered by the distributor. If the distributor has tried to deliver a package 3 times without success, the package will be removed from the distribution order and recorded as an incident. The function must always show for each processed package the package id and a message informing of it status: correctly delivered, pending (with the number of attempts made) or removed. The process must continue while the distributor still has packages to deliver.
- Write a function, **deliver**(), that simulates the delivery of packages from all distributors.
- Write a function, **removeDSMembr** (), that receives a distributor (identifier) and sets it as inactive. If the distributor has a pending package to be delivered, it must be reassigned to an active distributor that covers the area of the package. If no available distributor is found, the package will be registered as an incident.
- Create a Test function that allows you to test each of the functions described above.

Complexity Analysis

Each function should include a comment that indicates its temporal complexity, and briefly explain worst and best case.

Rules:

1. It is not allowed to use Python structures such as Lists, Queues or Dictionaries, etc.
2. However, you can use the implementations of SList or DList, which we have studied in class. If you want to use a stack or queue, its implementation must be based on a linked list (that is, internally the queue or stack cannot be implemented with a Python list).
3. To simplify the case study, in phase 1, it will not be necessary to represent or store any type of information regarding the date and time of the request or delivery of the orders.
4. The case study must be carried out by a group with two members (both must belong to the same teaching group). In no case will groups with more than two members be allowed. Individual groups are also not allowed since one of the competencies to be evaluated will be teamwork competency. If you don't have a partner, please send an email to your practice teacher.
5. Delivery method: A task entitled ‘Amazon 2020 Case Study’ will be published in the master group.
6. Delivery Date. May 4, 9.00 a.m.
7. Delivery format: a zip with the python (.py) scripts of the three phases. The name of the zip must be the two NIAS of the students, separated by a hyphen. For example, *10001234-10005678.zip*. Only one of the two members will be responsible for uploading the group solution.
8. Defense: during the class of Friday, May 8. The defense of the case study is an oral exam. Attendance is mandatory. If a student does not attend, his or her grade in the practical case will be 0. During this defense, the teacher will ask each team member a series of questions that must be answered individually. Each team member should be able to discuss any of the decisions made during the design or implementation of any of the functionalities described in the case study. The final grade of the case study will be conditioned by the grade obtained in the defense. If a student is not able to discuss and defend certain decisions, he/she will not be qualified with respect to them, even if they are correctly implemented.

9. It is recommended to follow the recommendations described in Zen of Python (https://www.python.org/dev/peps/pep-0020/) and the style guide (https://www.python.org/dev/ peps / pep-0008 /) published on the official Python website.
10. Your solution must be divided into subprograms (modules) in order to make it more readable, and easier to debug and maintain. Each module or subprogram must correspond to a well-defined task, and some modules will depend on others to operate.