Programming
Final exam demo task

The present mock exam was NOT designed to mimic the real exam as closely as possible. The task that you will be given on the final exam may differ significantly from the present one. However, this example may serve as a good starting point to revise the following key topics:

- Program flow constructs
- Functions and modules
- Data structures
- File input-output
- OOP (the solution may be refactored to OOP principles)

General requirements

- Split logic between multiple functions and optionally modules
- Apply the DRY (Don't Repeat Yourself) principle
- Read the dataset only once (at program startup)
- Use informative messages when user input is required
- Use proper coding style
- When using constant numeric values, don't put them into the code as numbers, extract them as named constants and reference them by name
- Add protection against incorrect user input

Task description

You will develop a simple prototype of a personal task / notification manager application. You are provided with a sample data file (tasks.txt), don't change its format (you can add new records if you wish). The information inside the file is organized as follows.

The first line contains **n** - the number of users of the application, then followed by **n** lines each describing a single user

Each user's description consists of three parts - user name, login and password hash separated by a semicolon. Read about hashing in a separate section below. If you are not fully sure how to apply the hashing logic, replace the hash sequences in the file with plain text passwords. The plain text passwords for the two default users are "johnpass" and "marypass" respectively (without quotes)

The next line contains **m** - the number of tasks created in the application, then followed by **m** lines each describing a single task

Each task consists of four parts: date and time stored as one single string in the YYYY-MM-DD HH:mm format, name of the task, its priority (Low/Medium/High) and the related user (user login is used to link a task to the user). Some of the tasks / notifications are related to all users, in this case the "all" string is used instead of a specific user login.

Your task / notification manager application should support the following capabilities:

1. Loading the entire dataset from a file
2. Saving the current dataset to a file
3. Simple user authentication by login / password pair
4. Showing a list of all tasks for the current user sorted in ascending order of the date
5. Creating new tasks
6. Deleting existing tasks
7. Logging out from the system

When showing a list of tasks for the current user include the tasks relevant to all users as well.

When creating / deleting tasks, the file with the dataset needs to be rewritten.

Storing passwords

Passwords are never stored or compared in plain (unencrypted) form, instead they are converted to a different representation using hash functions. E.g. if p1 and p2 are plain passwords, then we never compare p1 == p2, instead when authenticating a user the application checks whether hash(p1) == hash(p2).

A function for calculating password hash is given to you. A hash in our case is also a string.

Hints and advices

- Use debugging when necessary! It is your best help in most cases
- You are provided with a number of helper functions residing in the **utils** module:
  - **input_number** requests an integer number from the user until it falls in the specified range
  - **input_datetime** requests a date-time value in the YYYY-MM-DD HH:mm format and returns it as a string[1]
  - **input_value_from_list** requests a string from the user until it matches one of the allowed values
  - **get_hash** calculates a hash value of a plain password
- When reading a file it is advisable to use the **f.readline()** (where f is the file variable) function for this task rather than iterating over lines in a for loop, as you have to analyze individual strings with the number of users and the number of tasks (see file description above).

[1] Python offers a separate type for storing dates and time values but you can use string representation in this exam's task for simplicity

Program flow example

Below is only one of the possible program flows, you can make your own one provided that it meets the requirements stated above.

```
Enter username: john
Enter password: johnpass
1 - list all tasks
2 - add new task
3 - delete task
4 - logout
Select an action: 1
Date and time       Name                          Priority
2019-06-20 10:30    Programming exam              High
2019-06-30 23:55    End of year 1                 Medium
2019-07-12 10:45    Car technical inspection      Medium

1 - list all tasks
2 - add new task
3 - delete task
4 - logout
Select an action: 2
Enter task name: MDI second year starts
Enter date and time: 2019-09-01 09:00
Enter task priority (Low/Medium/High): Low
Do you want to create this task for all users? (Y/N): Y
Task added successfully

1 - list all tasks
2 - add new task
3 - delete task
4 - logout
Select an action: 1
Date and time       Name                          Priority
2019-06-20 10:30    Programming exam              High
2019-06-30 23:55    End of year 1                 Medium
2019-07-12 10:45    Car technical inspection      Medium
2019-09-01 09:00    MDI second year starts        Low

1 - list all tasks
2 - add new task
3 - delete task
4 - logout
Select an action: 4
Enter username: mary
Enter password: marypass

1 - list all tasks
2 - add new task
3 - delete task
4 - logout
Select an action: 1
Date and time       Name                          Priority
2019-06-30 23:55    End of year 1                 Medium
2019-07-11 08:30    Driving test                  High
2019-09-01 09:00    MDI second year starts        Low
```

```
1 - list all tasks
2 - add new task
3 - delete task
4 - logout
Select an action: 3
Your tasks:
1: End of year 1
2: Driving test
3: MDI second year starts
Select a task to delete: 3
Task deleted successfully

1 - list all tasks
2 - add new task
3 - delete task
4 - logout
Select an action: 1
Date and time       Name                          Priority
2019-06-30 23:55    End of year 1                 Medium
2019-07-11 08:30    Driving test                  High
```