# Introductory course in programming (DAT455)

15 March 2021

Examiner: Krasimir Angelov

**Note:** Solutions must be uploaded to Canvas as a single Python file. It is not necessary to comment on your code, it is more important that the code is clean and easy to read.

All help materials are allowed, but everyone is expected to work alone. Cooperation is not allowed and cheating may lead to suspension from the university.

The grading schema for DAT455 is G/U. You need at least 15 points to get grade G.

## Question 1

The martian year has 668.5907 sols, so just like on Earth, the martian calendar has leap years. A leap year is a year which is odd or divisible by ten. The leap year has 669 sols, while a normal year has 668. Write a program which asks the user for a year and prints out its length in days.

The following are three example dialogs:

```
Enter a martian year: 189
Year 189 has 669 sols

Enter a martian year: 218
Year 218 has 668 sols

Enter a martian year: 220
Year 220 has 669 sols
```

**(7 points)**

## Question 2

The command grep on Unix/Linux can search for a given word in a file. Implement the function:

```
def grep(word,file_name,exact_match):
    …
```

which would do the same (but in a simplified way). Here the argument `word` is what we are searching for, and the argument `file_name` tells us which file to open. The function should print the line numbers and the lines themselves that contain the given word. When `exact_match` is `True`, the function should look only for words that exactly match the search word. Otherwise, it also includes words which start with the given word.

All words in the file are assumed to be separated by spaces, so you can safely use `split()` to separate each line to words.

Examples:

```
>>> grep("Python","example.txt",True)
1: Python is an interpreted …
5: Python is dynamically-typed and …
7: Python is often described as …

>>> grep("Python","example.txt",False)
1: Python is an interpreted …
2: Python's design philosophy …
5: Python is dynamically-typed and …
7: Python is often described as …
```

**(10 points;** *7 points if you choose to ignore argument exact_match***)**

## Question 3

Implement the class `MessageBox` which makes it possible to send messages between users. It should have the following methods:

- An `__init__` method which initializes the state of the object to contain no messages.
- A method:

  ```
  def sendMessage(self, sender, receiver, message): …
  ```

  which sends a message from the sender to the receiver. The first two arguments are the user names of the sender and the receiver. The last argument is the text of the message itself.
- A method:

  ```
  def readMessages(self, user): …
  ```

  returns a list of unread messages sent to the given user name. Once the method returns the messages, they are all considered read. A second call to the method should not return them again. On the other hand, it may return more messages, if sendMessage was called in the meantime.

An example use of the class:

```
>>> box = MessageBox()
>>> box.sendMessage("john", "mary", "hi")
>>> box.sendMessage("john", "mary", "going for lunch?")
>>> box.sendMessage("john", "greg", "did you finish the
book?")
>>> box.readMessages("mary")
["hi", "going for lunch?"]
>>> box.readMessages("mary")
[]
>>> box.sendMessage("john", "mary", "sorry, I will be
late")
>>> box.readMessages("mary")
```

```
["sorry, I will be late"]
>>> box.readMessages("greg")
["did you finish the book?"]
>>> box.readMessages("john")
[]
```

**(13 points)**