Examination Advanced R Programming

Linköpings Universitet, IDA, Statistik

Course code and name:	732A94 Advanced R Programming
Date:	2022/10/24, 14–18
Teacher:	Krzysztof Bartoszek phone 013–281 885
Allowed aids:	The extra material is included in the zip file **exam_help_material_732A94.zip**
Grades:	A= $[18-20]$ points
	B= $[16-18)$ points
	C= $[14-16)$ points
	D= $[12-14)$ points
	E= $[10-12)$ points
	F= $[0-10)$ points
Instructions:	Write your answers in an R script file named **[your exam account].R** The R code should be complete and readable code, possible to run by copying directly into a script. Comment directly in the code whenever something needs to be explained or discussed. Follow the instructions carefully. There are **THREE** problems (with sub–questions) to solve.

Problem 1 (5p)

a) (3p) Explain what it means when a package is listed under Depends, Imports and Suggests in the DESCRIPTION file.
b) (2p) If you would be submitting a package to CRAN, explain how you would include in it a call to a function that comes from a package that is not on CRAN (but e.g. on GitHub only).

Problem 2 (9p)

READ THE WHOLE QUESTION BEFORE STARTING TO IMPLEMENT! Remember that your functions should **ALWAYS** check for correctness of user input!

a) (2p) In this task you should use object oriented programming in S3 or RC to write code that for a smart home medical cabinet. It should contain information on each medical component stored, for what symptoms it can be applied for, maximal dosage, how often it can be used, and when was the last time it was dispensed. Your goal is to first initialize the cabinet. Depending on your chosen OO system you can do it through a constructor (RC) or by implementing a function `create_medical_cabinet()` (S3). The constructing function should take one argument, `max_number`, the maximum number of medications that can be stored. The cabinet object should contain for each medication the information describing it, in particular a unique id of it, a text name (you may have the same one e.g. `"a"` for every medication), the symptoms it can be applied for (you may have the same one e.g. `c("a","b")` for every medication), an number for maximal dosage, its last usage time (this can be either an integer representing the day of the year or more ambitiously e.g. a `Date` (see `?Date`) class object), after how much time (you choose how to represent time) the medication can be used again, and how much of it is stored. Provide some example calls to your code.

```
## example call to create a cabinet object
my_cabinet <- create_cabinet(max_number=100) # S3
my_cabinet <- cabinet$new(max_number=100) # RC
```

b) (3p) Now implement a function called `add_medication_to_cabinet()` that allows one to add new medicine to the cabinet. The function should have five parameters: the text name of the substance, a text vector of symptoms it can be applied for, maximal dosage, how often it can be used, and the amount of it. The id is to be automatically generated. Do not forget that the cabinet has a maximum capacity! Provide some example calls to your code.

```
## S3 and RC example calls
my_cabinet <-add_medication_to_cabinet(my_cabinet,"onion_sugar_syrup",
c("cough","sneezing"),2,1,5)

## if using RC you may also call in this way
my_cabinet$add_medication_to_cabinet("onion_sugar_syrup",c("cough","sneezing"),2,1,5)
```

c) (3p) Now implement a function called `dispense_medication()` that for a given symptom, and requested dosage provides a medication The function should take three parameters, the current day (or other chosen by unit of current time, can be just an number), symptom(s) (you can make it easier and just provide a single symptom here, not a vector of them), and requested dosage (can be just a number). The the smart cabinet has to provide a medication based on the symptom (it will be enough that the symptom is inside the vector of symptom for a medication). The smart cabinet should react appropriately if the requested dosage exceeds the maximal one, if not enough time has passed between requests, or there are not enough doses in the cabinet. Remember that dispensing medicine reduces the amount of it. Provide some example calls to your code.

```
## S3 and RC call
my_cabinet<-dispense_medication(my_cabinet,"cough", 1, 1)

## if using RC you may also call in this way
my_cabinet$dispense_medication("cough", 1, 1)
```

d) (1p) Implement a function that displays the state of the cabinet. You are free to choose yourself how to report the state! This function has to also work directly with `print()`.

```
# calls to show state of cabinet
my_cabinet; print(my_cabinet)
```

Problem 3 (6p)

a) (3p) You are given a vector of integers of which you know that every number except one of them appears twice. That number appears only once. Implement a function (you may **not** use the algorithm provided in the file `find_singleton.R`) that takes as its input a vector of integers and finds the singleton number. Do not forget that your function should check for correctness of input and react appropriately. You need to check for integer input, but you can assume that the all numbers except one appear twice, and that the singleton number exists.
b) (1p) What is the (pessimistic) computational complexity of your implementation in terms of the input vector's size?
c) (1p) In the file `find_singleton.R` you are provided with an algorithm that solves this problem. What is its (pessimistic) computational complexity in terms of the input vector's size?
d) (1p) Implement a unit test that compares your implementation with the `f_findsingleton()` function provided in the file `find_singleton.R`.