Examination Advanced R Programming

Linköpings Universitet, IDA, Statistik

Course code and name:	732A94 Advanced R Programming
Date:	2022/11/30, 8–12
Teacher:	Bayu Beta Brahmantio
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) Provide ways in which your R code's speed can be improved. Explain why each proposed change improves speed.
b) (2p) If a package is listed under Suggests, what is the correct way of calling a function from it inside your implemented package? Is it sufficient to call it in the same way as a function from a package listed under Depends or Imports? If not what else needs to be done and why?

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 printer maintains the paper status. The printer contains a number of paper types, and it should be remembered how much of each paper type is present and what is the maximum amount of each type of paper. Furthermore the history of the printing should be remembered. A printing event consists of a timestamp, number of printed pages and their type, file name, status (success/failure), and (system, i.e. you) generated unique id. Your goal is to first initialize the printer management system. Depending on your chosen OO system you can do it through a constructor (RC) or by implementing a function `create_printer_manager()` (S3). The constructing function should take one argument, `v_max_amount_paper`, a vector of numbers giving, the maximum amount of paper that can be stored for each paper type. The printer manager object should contain for each paper type the maximum amount of paper that can be stored and how much is currently present. Furthermore, it should store an object that records the printing history. Notice, that it requires that each paper type has some id. You decide about the timestamp's type, it can be a number, `POSIXct` object, e.t.c. Remember that the amount of paper is an integer. Provide some example calls to your code.

```
## example call to create a printer management object
my_printer_manager <- create_printer_manager(v_max_amount_paper=c(500,500,500)) # S3
my_printer_manager <- printer_manager$new(v_max_amount_paper=c(500,500,500)) # RC
```

b) (3p) Now implement a function called `add_paper()` that allows one to add paper to the printer. The function should have one parameter: a vector of numbers of each type of paper for adding to the printer. Do not forget that there is a maximum capacity for each paper type, and that the paper types are already defined! Provide some example calls to your code.

```
## S3 and RC example calls
my_printer_manager <-add_paper(my_printer_manager,c(200,200,200))


## if using RC you may also call in this way
my_printer_manager$add_paper(c(200,200,200))
```

c) (3p) Now implement a function called `print_document()` that sends a document for printing.

The function should take three parameters, a timestamp, a file name, and how many pages of each paper type is needed for printing. The printer manager should react accordingly if it can or cannot print. Then, it should update the printing history and paper status Provide some example calls to your code.

```
## S3 and RC call
my_printer_manager<-print_document(1,"f1.pdf", c(10,0,1))

## if using RC you may also call in this way
my_printer_manager$print_document(1,"f1.pdf", c(10,0,1))
```

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

```
# calls to show paper status of he printer
my_printer_manager; print(my_printer_manager)
```

Problem 3 (6p)

Autobiographical numbers are very special numbers in the sense that they describe themselves. Its first digit (in the decimal system) says how many 0s the number has, the second how many 1s, the third how many 2, e.t.c. There are actually very, very few autobiographical numbers. In fact, these are all of them 1210, 2020, 21200, 3211000, 42101000, 521001000, 6210001000 (entry A046043 of the Online Encyclopedia of Integer Sequences) **a) (3p)** Implement a function that takes an integer input and verifies if the number is an autobiographical number. You **may not** directly check for equality with the values from the provided set of these numbers. Do not forget to check for correctness of input.
b) (2p) What is the computational complexity of your implementation in terms of the number of digits of the passed integer?

d) (1p) Implement a unit test that compares your implementation with a direct check against the provided set of autobiographical numbers.