Examination Advanced R Programming

Linköpings Universitet, IDA, Statistik

Course code and name:	732A94 Advanced R Programming
Date:	2021/10/25, 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) When implementing a shiny app what are the two fundamental functions that should be implemented? Discuss the role of each one.
b) (2p) What is the `reactive()` function used in a shiny app for? In what situations can it be important?

Problem 2 (10p)

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 keeps track of plants planted in a garden. It should contain information on the plant, when it was planted and when should it be harvested.

Your goal is to first initialize the garden. Depending on your chosen OO system you can do it through a constructor (RC) or by implementing a function `create_garden()` (S3). The constructing function should take one argument, `max_number_plants`, the maximum number of plants that can be grown. The garden object should contain for each plant information describing it, in particular a unique id of the plant, a text name (you may have the same one e.g. `"a"` for every plant), the date it was planted (this can be either an integer representing the day of the year or more ambitiously e.g. a `Date` (see `?Date`) class object) and after how much time (you choose how to represent time) the plant should be harvested. Provide some example calls to your code.

```
## example call to create a garden object
my_garden <- create_garden(n=100) # S3
my_garden <- garden$new(n=100) # RC
```

b) (3p) Now implement a function called `add_plant_to_garden()` that allows one to add a new plant to the garden. The function should have three parameters: the text name of the plant, its planting date, and after how much time it should be harvested. The id is to be automatically generated. Do not forget that the garden has a maximum capacity! Provide some example calls to your code.

```
## S3 and RC example calls
my_garden <-add_plant_to_garden(my_garden,"strawberry","2021-04-15",60)
my_garden <-add_plant_to_garden(my_garden,"strawberry",95,60)

## if using RC you may also call in this way
my_garden$add_plant_to_garden("strawberry",85,60)
```

c) (3p) Now implement a function called `harvest_plants()` that for a given date harvests all plants that should be harvested. The function should take one parameter, the day in question. It should harvest (i.e. remove from the garden object) all plants that have a harvest date equal or earlier than the provided date. The function should return the number of harvested plants. Remember that there could be nothing to harvest. Provide some example calls to your code.

```
## S3 and RC call
my_garden<-harvest_plants(my_garden,"2021-07-01")
my_garden<-harvest_plants(my_garden,172)


## if using RC you may also call in this way
my_garden$harvest_plants(172)
```

d) (2p) Implement a function that displays the state of the garden. 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 garden
my_garden; print(my_garden)
```

Problem 3 (5p)

a) (2p) It is often important to change from a binary representation of an integer to the integer itself. The following (**extremely** ineffective) algorithm does this. The input is a character vector, `v`, (consisting of `"0"`s and `"1"`s).

```
1: procedure BitsToInt(v)
2:     int_value=0
3:     j=0
4:     for i = length(v) ... 1 do
5:         if v[i]=="1" then
6:             for k = 1 ... 2^j do
7:                 int_value=int_value+1
8:             end for
9:         end if
10:        j=j+1
11:    end for
12:    return int_value
13: end procedure
```

Implement a function that takes as its input a character vector of `"0"`s and `"1"`s and calculates the corresponding integer value according to the pseudocode above. Do not forget that your function should check for correctness of input and react appropriately.

b) (1p) What is the (pessimistic) computational complexity in terms of the number of binary digits of the number?

TIP: Recall the sum of a geometric progression, $a \neq 1$,

$$\sum_{i=0}^{n-1} a^i = \frac{1-a^n}{1-a}.$$

c) (2p) Implement a unit test that compares your implementation with direct calculation of the value using `compositions::unbinary()`. As `compositions::unbinary()` takes a character string as input and not a vector, use e.g. `paste0(my_binary_vector,collapse="")` to change your input vector to a character string.