Examination Advanced R Programming

Linköpings Universitet, IDA, Statistik

Course code and name:	732A94 Advanced R Programming
Date:	2024/02/28, 8–12
Teacher:	Krzysztof Bartoszek
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 R scripts named according to the pattern **[your exam account]_*.R** The R code should be complete and readable code, possible to run by calling `source()` directly on your `*.R` files. Comment directly in the code whenever something needs to be explained or discussed. Follow the instructions carefully. There are **TWO** problems (with sub–questions) to solve.

Problem 1 (11p)

READ THE WHOLE QUESTION BEFORE STARTING TO IMPLEMENT! Remember that your functions should **ALWAYS** check for correctness of user input! For each subquestion please provide **EXAMPLE CALLS!**
You are to implement an object that stores information on mathematical functions on integers.
a) (3p) In this task you should use object oriented programming in S3, S4 or RC to write code that handles mathematical functions with integer parameters. Depending on your chosen OO system you can do it through a constructor or by implementing a function `create_function_handler()`. The constructing function should not take any arguments. The object should contain for each function, the information concerning it, in particular a unique id of it; the function itself (an `R` function); whether it converges or not; and a cache for function values for different values of the integer argument. The `function_handler` object should also store the total number of functions, how many are convergent, how many not.

```
## example call to create a function handler object
my_functions <- create_function_handler() # S3
my_functions <- function_handler$new() # RC
```

b) (4p) Now implement a function called `add_function()` that adds a function to your function handler object. The function should have two parameters: the function on integers and a logical variable if it is convergent or not.

```
## S3 and RC example calls
fHn1<-function(n){sum(1/(n:1))}
fHn2<-function(n){sum(1/((n:1)^2))}
my_functions <-add_function(my_functions,fHn1,FALSE)
my_functions <-add_functions(my_functions,fHn2,TRUE)
## if using RC you may also call in this way
my_functions$add_function(my_functions,fHn1,FALSE)
my_functions$add_function(my_functions,fHn1,TRUE)
```

c) (3p) Now implement a function called `get_function_value()` that calculates the value of the function identified by a specific identifier for a particular integer number. This calculated value should be provided to the user and also stored in some cache structure so that it does not need to be recalculated. In particular, if a value is cached, then your implementation should not recalculate it. The function `get_function_value()` should take two arguments, the unique id of the function and the integer value at which it will be evaluated.

```
## S3 and RC example call
n<- 10
my_functions<-get_function_value(my_functions,1,n)
#but you are free to have a different id system
## if using RC you may also call in this way
my_functions$get_function_value(1,n) #but you are free to have a different id system
```

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

```
# calls to show your functions
my_functions; print(my_functions)
```

Problem 2 (9p)

a) (2p) Consider the recursive sequence defined as

$$V_{n+1} = aV_n + bV_{n-1}$$

with initial conditions $V_0 = 2$, $V_1 = a$. Please implement a **recursive** function that takes as input n, a, and b and calculates V_n. Your implementation has to be the direct recursion as above, without any improvements (these will come in later questions). Run the code and time it, use, e.g., `system.time()`, for different values of n (do **NOT** take n greater than 30, as this will take too long to run). Report the running times for different n.
Please provide **EXAMPLE CALLS** to your function.

b) (1p) What is the computational complexity of your code?

c) (2p) Discuss how the computational complexity and hence running time of your solution can be improved.

d) (2p) Please implement the improvements, and run the code and time it, use, e.g., `system.time()`, for different values of n (do **NOT** take n greater than 30, as this will take too long to run). Report the running times for different n.
Please provide **EXAMPLE CALLS** to your function.

e) (1p) What is the computational complexity of your improved code?

f) (1p) It can be mathematically shown that the following properties hold $V_{2n} = V_n^2 - 2b^n$ and $V_{2n+1} = V_n V_{n+1} - ab^n$. Implement unit tests based on these properties.