Homework 1

Rmarkdown and Elementary programming

Kun Zhou, uid 204688165

You are required to write a *markdown document* containing text, code, and at least one example. Thus you must create an Rmd file, that RStudio can convert to html.

You may use `R` functions you have written in one exercise in your subsequent exercise solutions, as a natural common-sense alternative to copy-pasting your lines of code.

1. Using only simple logical expressions, write an `R` function, for example named `my_floor`, that accepts a floating-point number and returns the greatest integer number that is less than or equal to the argument floating-point number.

Using `%/%` can get the quotient of integer division. So to get the floor, just apply integer division to the floating-point by 1.

```
my_floor <-function(x)
{
  return(x%/% 1)
}

my_floor(4.2)
```

```
## [1] 4
```

```
my_floor(-4.2)
```

```
## [1] -5
```

```
my_floor(0)
```

```
## [1] 0
```

2. Using only simple logical expressions and/or your floor function, write an `R` function, for example named `my_ceil`, that accepts a floating-point number and returns the smallest integer number that is greater than or equal to the argument floating-point number.

For any floating-point number, determine whether it is an integer. If it is an interger, return the integer, otherwise use `my_floor` first and then add 1.

```
my_ceil <- function(x)
{
  y = x %/% 1
  if (x == y)
    return(y)

  else
    return(y+1)
}
my_ceil(4.2)
```

```
## [1] 5
```

```
my_ceil(-4.2)
```

```
## [1] -4
```

```
my_ceil(0)
```

```
## [1] 0
```

3. Using only simple logical expressions and/or your `R` functions, write a function that accepts a floating-point number x and an integer-valued parameter k that returns the floating-point number rounded to k decimals. A negative value of k corresponds to rounding to a number with k trailing zeroes.

Multiply 10^k with the given number if the number is positive, then round it to a integer. Finally divide the integer with 10^k.(Notice: k can be negative interger) If the given number is negative, round its absolute value and then add the negative sign.

```
round_k <- function(x, k)
{
  if (x>=0)
  {
    y = x * 10 ^ k
  z = (y %% 1) * 10 # get the tenth digit and round y according to it
  if (z < 5)
    return(my_floor(y) / 10 ^ k)
  else
    return(my_ceil(y) / 10 ^ k)
  }
  else
    return(-round_k(-x, k))
}
round_k(1.234, 2)
```

```
## [1] 1.23
```

```
round_k(1.236, 2)
```

```
## [1] 1.24
```

```
round_k(-15, -1)
```

```
## [1] -20
```

```
round_k(-14, -1)
```

```
## [1] -10
```

```
round_k(4.4, 0)
```

```
## [1] 4
```

4. Using only simple logical expressions and/or your `R` functions, write a function that accepts a non-negative integer number x and a positive integer k that returns x modulo k.

`%%` returns the remainder of the integer division.

```
mod <- function(x, k)
{
  return(x %% k)
}

mod(10, 3)
```

```
## [1] 1
```

```
mod(10, 2)
```

```
## [1] 0
```

5. *Bubble sort* is a simple algorithm that sorts a vector or numbers by evaluating consecutive pairs of numbers and swapping them if the former number in the pair is greater than the latter. Implementing this algorithm, write an `R` function, for example named `my_bubble` that accepts a vector of floating-point numbers and returns the vector sorted from the smallest number to the largest.

The Question has explained the principle of bubble sort in detail. For bubble sort, it is an algorithm with $O(n^2)$, so merge sort is a better alogrithm with $O(n \log n)$

```
my_bubble <- function(x)
{
  n = length(x)
  for (i in 1:(n-1))
  {
    for(j in 1:(n-i))
    {
      if (x[j] > x[j+1])
      {
        temp = x[j]
        x[j] = x[j+1]
        x[j+1] = temp
      }
    }
  }
  return(x)
}
my_bubble(5:1)
```

```
## [1] 1 2 3 4 5
```

```
my_bubble(c(3,5,4,1,2))
```

```
## [1] 1 2 3 4 5
```

6. Suppose x and y are two non-zero double-precision floating point numbers, one very large and the other very small. Explain why `R` under some circumstances can return (erroneously) that the identity $x + y = x$ is true.

According to R document, there are also denormal(ized) (or subnormal) numbers with absolute values above or below the range given above but represented to less precision. A large number may have less precision in demicals and a small number is out of the precision of the large number. So when they are added up, the large one neglects the small one.

```
x = 10^10
y = 10^-10
x == x+y
```

```
## [1] TRUE
```

Submit

Upload your homework before the end of the due date via the *Homework* page on the class CCLE. Submit only the Rmd file, but make sure RStudio knits it correctly to html.

Cooperation

A limited amount of cooperation on the homework is allowed, but everybody hands in their own document. If you have cooperated in a group, indicate this in your document and mention the names of the other students in the group. Groups of more than four students are not allowed. Everybody in a group will get the same grade, even if the individual documents are different.