Econometrics 1 *Applied Econometrics with R*

Lecture 6: Programming with R (3)

黄嘉平

中国经济特区研究中心 讲师

办公室：文科楼2613

E-mail: huangjp@szu.edu.cn

Tel: (0755) 2695 0548

Website: https://huangjp.com

Review

- Function

```
function_name <- function
(x) {
  ……
  ……
  return(…)
}
```

- Loop

```
for (variable in sequence) {
  command_A
  command_B
  ……
}
```

- Condition: the `if` statement

```
if (condition) {
  command_A
  command_B
  ……
} else {
  command_C
  command_D
  ……
}
```

Practice: random walk

- A stochastic process $Z(t)$ is called a random walk if

$$Z(0) = 0 \quad \text{and} \quad Z(t) = Z(t\text{-}1) + B \quad \text{for } t = 1, 2, \ldots,$$

  where B is a random variable that takes value 1 with probability p and -1 with probability $(1\text{-}p)$. Generate a random walk with 200 periods (e.g. $t = 0, \ldots, 200$) and $p = 0.6$, and plot your results.

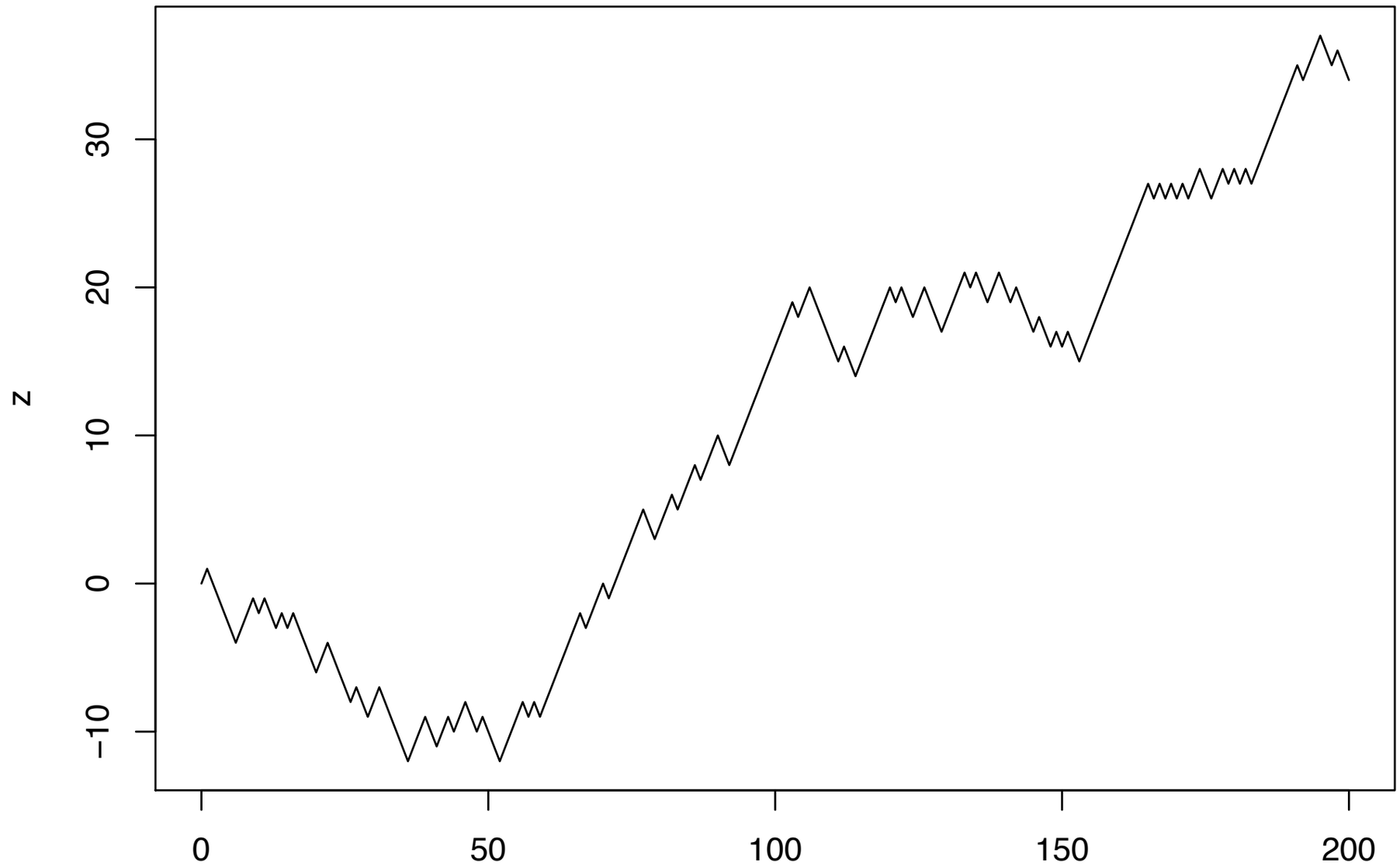
- Use `rbinom()`

Practice: random walk

```
z <- rep(0, 201)
for (i in 1:200) {
  if (rbinom(1,1,0.6) == 1) {inc <- 1}
  else {inc <- -1}
  z[i+1] <- z[i] + inc
}

plot(0:200, z, type="l", xlab="")
```

Practice: random walk



Recursion (递归)

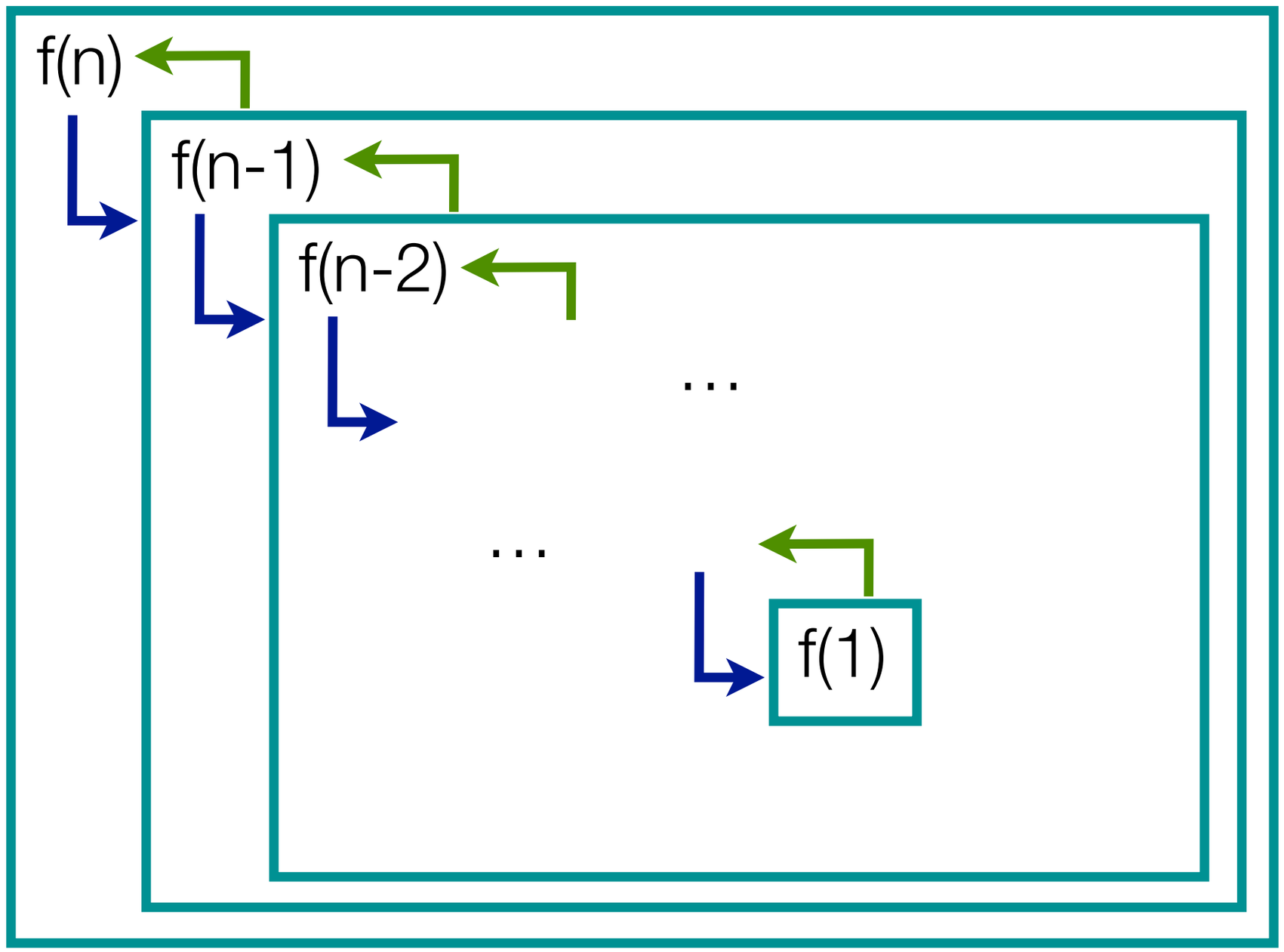
- A recurrence relation (递推关系) of the factorial function

$$f(n) = n! \quad \Leftrightarrow \quad f(n) = n \times f(n-1) \text{ with } f(1) = 1$$

- Calculate factorial using recursion:

```
facrec <- function (n) {
   if (n <= 1) {
      return(1)
   } else {
      return(n * facrec(n-1))
   }
}
```

Recursion

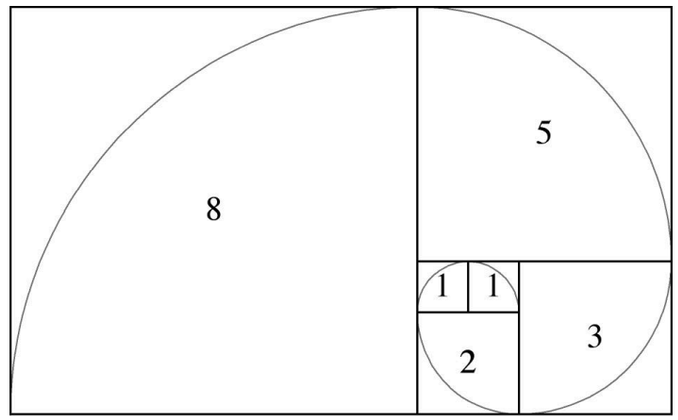


Practice: the Fibonacci numbers

- The Fibonacci sequence (or Fibonacci numbers) is

  1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, …

  or mathematically



$$F_n = F_{n-1} + F_{n-2} \text{ with } F_2 = F_1 = 1$$

- Write a functions `fib(n)` which returns the *n*th Fibonacci number F_n. Use recursion in your program.

Practice: the Fibonacci numbers

```
fib <- function (n) {
  if (n == 1 | n == 2) {
    f <- 1
  } else {
    f <- fib(n-1) + fib(n-2)
  }
}
```

Write a program

Practice: find the maximum

- Define a function `maxfun` which returns the maximum number of a set of given real numbers.

- For example, if we let

  ```
  > x <- c(3, -2.5, 0, 34, pi)
  ```

  then it is expected `maxfun(x) = 34`

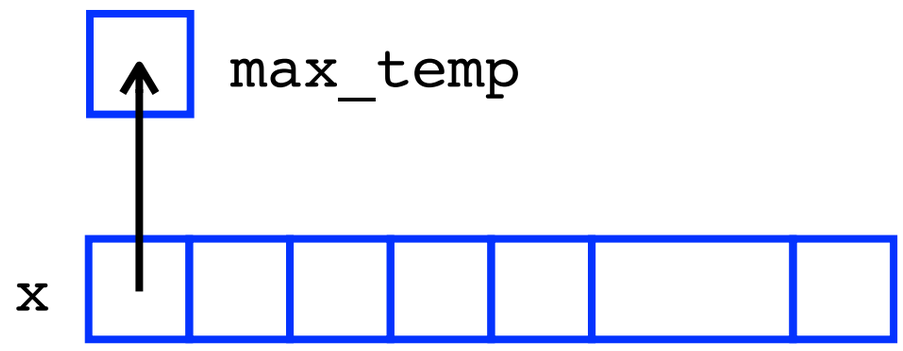
- Note: your program must take care of the case that `x` contains only one number.

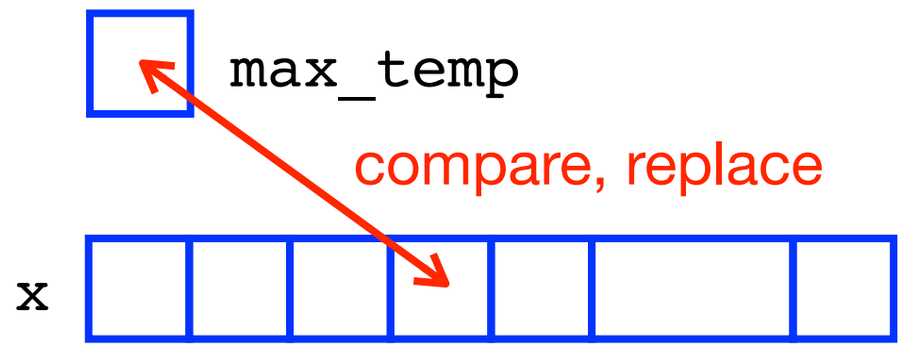
Practice: find the maximum

```
maxfun <- function (x) {
  n <- length(x)

  max_temp <- x[1]

  for (i in 1:n) {
    if (x[i] > max_temp) {
      max_temp <- x[i]
    }
  }
  return(max_temp)
}
```

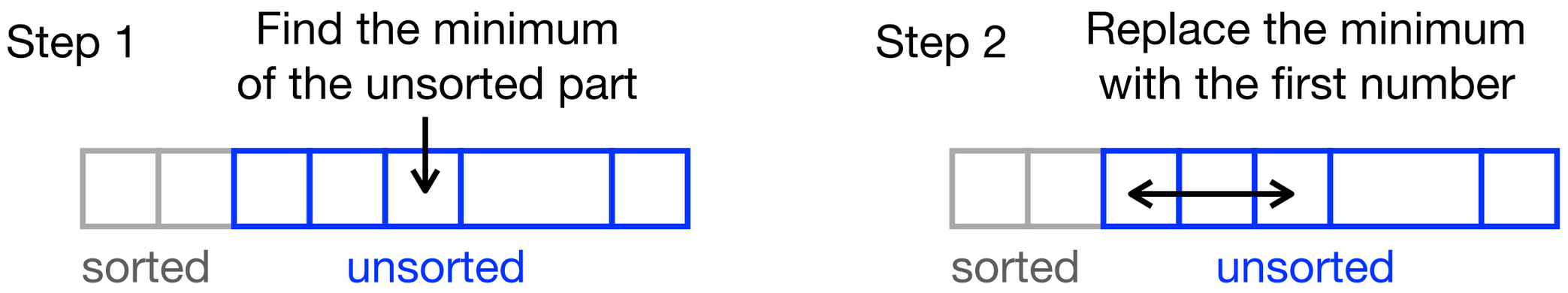




Practice: sort a list of numbers

- In many situations, we want to *sort* a list of numbers in ascending (or descending) order, i.e., from the smallest to the largest (from the largest to the smallest).

- Selection sort (ascending order):



Practice: sort a list of numbers

- Write a function `selesort` which implement the selection sort in ascending order. Apply it to previously defined `x`

- Hint:

1. You need to find both the *value* and the *location* of the minimum of the unsorted part.

2. The output (the `return` part) of a function can be a vector.

Practice: sort a list of numbers

```
minfun <- function (x) {
   n <- length(x)
   min_temp <- x[1]
   loc_temp <- 1
   for (i in 1:n) {
      if (x[i] < min_temp) {
         min_temp <- x[i]
         loc_temp <- i
      }
   }
   return(c(min_temp, loc_temp))
}
```

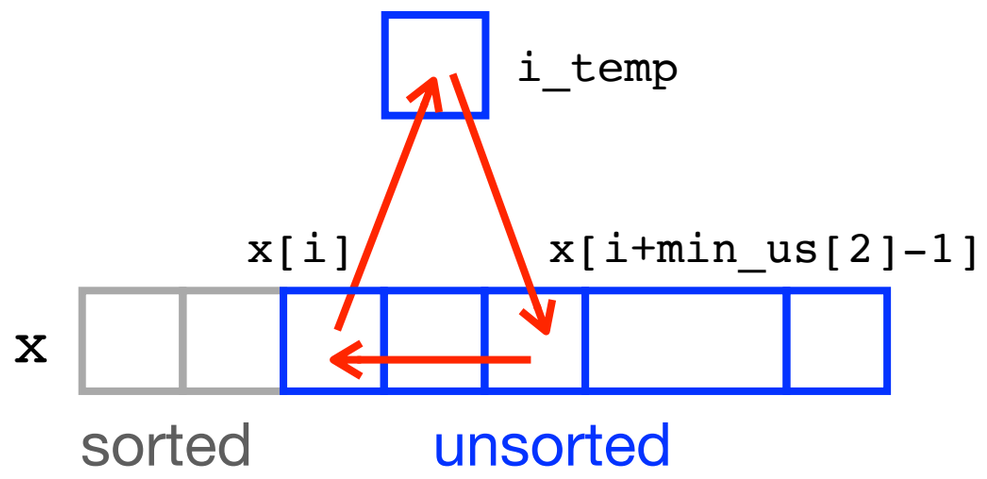
Practice: sort a list of numbers

```
selesort <- function (x) {
  n <- length(x)
  for (i in 1:n) {
    unsorted <- x[i:n]
    min_us <- minfun(unsorted)
    i_temp <- x[i]
    x[i] <- min_us[1]
    x[i + min_us[2] - 1] <- i_temp
  }
  return(x)
}
```

Specify the unsorted part

Find the minimum

Swap



Assignment 1

Assignment 1

- Learn the *insertion sort* algorithm from Wikipedia (or other websites): https://en.wikipedia.org/wiki/Insertion_sort

- Write a program that sorts a given sequence in **descending** order which meets the following conditions
    1. use insertion sort algorithm to write a function named `inssort()`
    2. you should not use `while` loop
    3. your function should print all partially sorted sequences, one in a line, e.g.,
       ```
       > inssort(c(3,5,1,4,2))↩
       [1] 3 5 1 4 2
       [1] 5 3 1 4 2
       [1] 5 3 1 4 2
       [1] 5 4 3 1 2
       [1] 5 4 3 2 1
       ```
    4. generate a sequence with 10 positive integers and use the above function to sort it, and save your sorted sequence in a new variable

- Save your program in an .R script file and submit it by email before 2018-11-06 19:00. (Note: write your email in a good manner!!)