Project 1

Due Friday, April 7th, 23:59

General notice about submissions

In every problem, we ask you to create one or more functions with specific names. Suppose you are asked to create functions named foo() and bar(). When you submit your code, the code must have **ONLY** the definitions of the functions, and not any **CALLS** to them. Also, you **must not include** any **set.seed(), print()** or **cat()** statements in your submissions.

Good example:

```
# group1_project1.R
foo <- function(x){
  return(2*x)
}
bar <- function(x, y){
  return(x+y)
}
baz <- function(N){
  return(N*2)
}
```

Bad example:

```
# mysubmission.r      # Name of the R script is incorrect

foo <- function(x){
  return(2*x)
}

foo(c(1,2,3))         # don't call the function here

bar <- function(x, y){
  return(x+y)
}

mysum <- bar(3,2)     # no assignment, no function call.

set.seed(111)         # don't set the random seed here

baz <- function(N){
  print(N)            # don't use print or cat statements
  runif(N)*2          # function should have a return statement
}
```

The reason is that when we run your code for testing purposes, such calls may create some unwanted side effects and interfere with the grading.

When you work on your programs, feel free to call your functions to check that they give the correct results. However, in the final submission **keep only the function definitions. Your submission should not do anything except for defining the functions. Do not call the functions.**

We are going to test your programs with new, undisclosed input along with the examples given here. Write your programs to be as general as possible within the problem description. We suggest that you invent your own test cases for the problem, figure out what output you expect, and verify that your code gives the expected output.

When you are ready to submit, clear your environment in RStudio (the sweep button in the Environment tab), or click on Kernel->Restart Kernel in your Jupyter Lab/Notebook and do a final check. By refreshing the environment, you erase existing variables, which may confound your program. Try out your code one last time with a fresh environment before deleting all the function calls. Afterward, **delete the function calls and all prints**. **Only function definitions** (without any prints to the console) **should remain** in your R Script.

You must work on the assignment as a group. Your code will be checked by an artificial intelligence-based algorithm for cheating. If cheating is detected in your code in the submitted assignments, you may immediately fail the course.

Late submissions can be accepted with a penalty of 20 points per day (that is, if you submit a perfect homework one day late, you will get only 80. If two days late, only 60, etc.).

QUESTION 1 (15 points)

Ann is hosting a pancake party for her friends. She is planning to invite *N* number of friends. Using the following information, calculate the price of cooking a pancake meal for Ann's friends in TL.

To cook pancakes for **4 people**, you need to use the following ingredients:

- 500ml milk
- 2 eggs
- 200g flour
- 30g butter
- 30g sugar
- Half a jar of honey as a topping

On average, the prices of products in the supermarket are as follows:

- 1 liter milk: 25 TL
- A pack of 10 eggs: 30 TL
- 2kg flour: 40 TL
- 125g butter: 40 TL
- 1kg sugar: 90 TL
- A jar of honey*: honey_price* TL

The price of honey is constantly changing. Therefore, you will take the price of a jar of honey as an argument of the function (*honey_price*).

Write a function named *pancake_meal(N, honey_price*) that calculates the price of cooking pancakes for *N* number of people, given the current jar of honey price. **Return the price** from the function **without rounding**.

- The function should take two arguments: *N* and *honey_price*

- The function should return a ***numeric*** variable that contains the price of the meal for *N* people for the given *honey_price.* Don't round the price.

Note: If you need to cook pancakes for N people, calculate the price of making the pancakes for N people, not the price of buying the products as a whole. For example, if you need to cook pancakes for 8 people, you would need 4 eggs. The price of the eggs needed for 8 people is: (30 / 10) x 4 = 12. Apply this logic to other ingredients.

Examples

```
> pancake_meal(10, 70)

174.5

> pancake_meal(5, 50)

74.75

> pancake_meal(1, 100)

21.2
```

QUESTION 2 (15 points)

You are working as an analyst for a company. Your company is making a preliminary analysis of the financial performance of a major production studio to create an annual report for the industry. Given the information below on the annual Box Office Grosses (million $), you will find the performance change between the two requested years.

2013	**2014**	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**	**2022**
4167	3140	3587	5043	3917	4901	5123	6410	6398	7891

The formula for performance change is as follows:

Performance change (%) = (New year – Old year) / Old year x 100

Write a function called *performance_change(old_year, new_year)* that calculates the performance change (%) between an old year and a newer year. **Round** the performance change to **two decimal places** (e.g. 23.45).

- The function should take two arguments: *old_year* and *new_year*
- The function should return a ***numeric*** variable rounded to **exactly two** decimal places (e.g. 23.45). If there are fewer or more than two decimal places, you may lose points.

Hint 1: Inside the function, create a vector that contains the annual gross values. The names of the vector will be the years.

Hint 2: If the result comes with the year name as a vector, you can use the ***unname*** function to only return the numeric result.

Hint 3: Use the round() function to round the performance change to two decimal places.

Examples

```
> performance_change("2021", "2022")

23.34

> performance_change("2015", "2021")

78.37

> performance_change("2014", "2016")

60.61
```

QUESTION 3 (15 points)

Yusuf works at the water quality control department of a firm. He tests the daily quality of water for the tap water of the city of Istanbul. If the rate of contamination for the tap water is **larger than or equal to** a certain percentage (*threshold*), he marks the tap water as dangerous.

Write a function *dangerous_tap_water(contamination_rates, threshold)* that takes the daily contamination rates and a threshold, and returns how many days have dangerous tap water.

- The function should take two arguments: *contamination_rates* as a **vector** that contains **numeric** values, and a **numeric** *threshold*
- The function should return a ***numeric*** value that indicates the number of dangerous tap water days within the *contamination_rates* vector

Examples

```
> dangerous_tap_water(c(1.3016, 1.2698, 0.7143, 0.1905, 1.4762, 1.1429, 0.3175, 0.5556,
0.9841, 0.381), 0.97)

5

> dangerous_tap_water(c(2.3636, 2.0909, 1.9697, 0.9091, 1.5758, 2.2424), 1.9697)

4
```

QUESTION 4 (25 points)

A group of friends wants to find out who is the tallest and shortest in the group. Barbara is studying computer engineering and instead of finding who is the tallest and who is the shortest directly, she prefers to write a function to find it.

Write a function named *tallest_and_shortest(heights)* that takes a vector containing people's heights as a parameter and returns a vector that contains the names of the tallest and shortest among the group. Assume that the group of friends will always contain more than one person.

- The function should take one argument: *heights* as a **vector** that contains **numeric** values

- The function should return a ***character vector*** that contains two elements: the name of the tallest person (with the name “tallest”), and the name of the shortest person (with the name “shortest”). Be careful about the order, first put the tallest, and then the shortest into the returned vector.

Examples

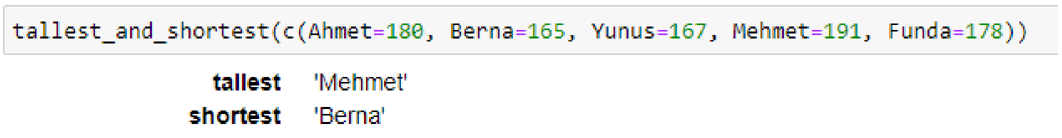
Vectors can appear differently on different platforms. Below is an example from Jupyter Notebook and RStudio.

```
> tallest_and_shortest(c(Ahmet=180, Berna=165, Yunus=167, Mehmet=191, Funda=178))

 tallest    shortest

"Mehmet"  "Berna"
```

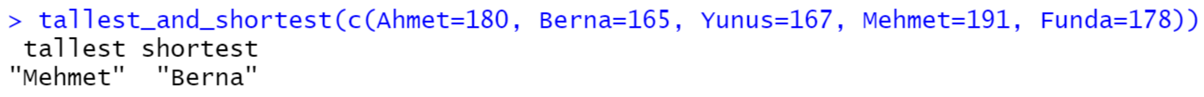
Jupyter Notebook:



```
tallest_and_shortest(c(Ahmet=180, Berna=165, Yunus=167, Mehmet=191, Funda=178))

   tallest   'Mehmet'
  shortest   'Berna'
```

RStudio:



```
> tallest_and_shortest(c(Ahmet=180, Berna=165, Yunus=167, Mehmet=191, Funda=178))
  tallest shortest 
 "Mehmet"  "Berna" 
```

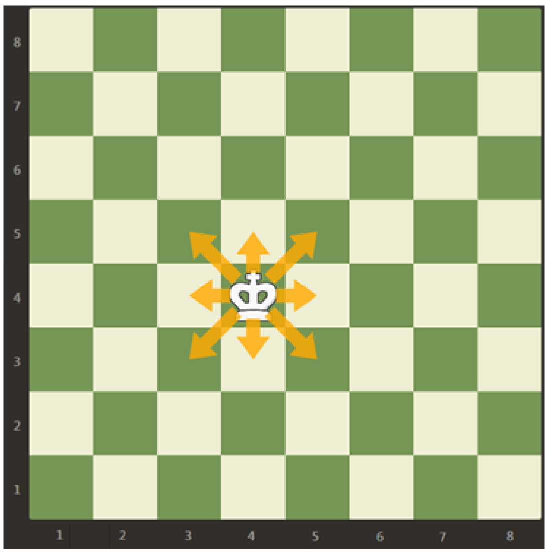
```
> tallest_and_shortest(c(Buse=180, Kaan=197, Ahmet=165, Burak=170, Selen=156, Melek=165))

  tallest   shortest

  "Kaan"  "Selen"
```

QUESTION 5 (30 points)

In the chess game, the king can move one square horizontally, vertically, or diagonally. Assume that there’s only one king on the chessboard (no other chess piece is present):



Find how many variants of the next move the chess king standing in the cell with given coordinates has.

Create a function called *chess_king(x, y)*, that takes the coordinates of the king as two arguments (horizontal row number *x* and vertical column number *y*), and returns a numeric value: the number of the moves possible moves for the king. The coordinates of the cell are given by two numbers from 1 to 8. First the row number (*x*), then the column number (*y*).

Hint: Since there are no other pieces on the board, you should pay attention to the corners. For instance, when the king is in the lower left corner (coordinates: x=1, y=1), it can't move to the bottom or the left, so it has only 3 possible moves.

Examples

chess_king(3, 2)

8

chess_king(1, 1)

3

chess_king(1, 2)

5

Submission

The function definitions should be submitted as a single source file named: ***groupN_project1.R***

So, if you are in group 7, every member in the group should submit an R script with the name: *group7_project1.R*

The R script should have the following contents:

```
pancake_meal <- function(N, honey_price) {
  # your code here
}

performance_change <- function(old_year, new_year){
  # your code here
}

dangerous_tap_water <- function(contamination_rates, threshold){
  # your code here
}

tallest_and_shortest <- function(heights){
  # your code here
}

chess_king <- function(x, y){
  # your code here
}
```

- The name of the submitted file should be ***groupN_project1.R*** **,** where **N** is your group number.
- You should only write the function definitions inside ***groupN_project1.R*** . Do not write any function calls inside ***groupN_project1.R*** .
- **Do not write** use any **prints/cats** in your code.
- Follow the guidelines under each question regarding the number of arguments of the function, and the type of variable to be returned from the function. You can use the template ***groupN_project1.R*** script to avoid erroneous function names or number of arguments.

Even if your code seems to work correctly, all of these conditions must be met to get a full grade. If you don't follow the submission rules, your code may not be successfully graded by the automatic grading system.

Contact your coach student assistants or teaching assistants (zeynep.yirmibesoglu@boun.edu.tr or sami.boz@boun.edu.tr) if you encounter any problems, or for anything you don't understand.