Koç University
COMP 125: Programming with Python
Homework #2

Deadline: April 4, Sunday at 23:59pm
Submission through: Blackboard
Make sure you read and understand every part of this document

This homework assignment contains 4 programming questions.

Download Hw2.zip from Blackboard and unzip the contents to a convenient location on your computer. Each file in Hw2.zip contains the starter code for one programming question.

Solve each question in its own file. **DO NOT CHANGE THE NAMES OF THE FILES. DO NOT CHANGE THE HEADERS OF THE GIVEN FUNCTIONS (FUNCTION NAMES, FUNCTION PARAMETERS). DO NOT USE TURKISH CHARACTERS.** If you want, you can add helper functions.

When you are finished, compress your Hw2 folder containing all of your answers. The result should be a SINGLE compressed file (file extension must be .zip, or .rar). Upload this compressed file to Blackboard.

Q1: Email Builder - 25 pts

Write a function build_email(firstname, lastname, entry_year) such that:
- firstname is the first name of the student (string)
- lastname is the last name of the student (string)
- entry_year is the year the student entered Koç University (string)

The function returns the e-mail address of the student, which is built by concatenating:
- First letter of the student's first name
- Student's last name in full
- Last two digits of student's year of entry
- "@KU.EDU.TR"

The returned e-mail address should be **completely in uppercase**.

You must check for the following erroneous inputs:
- First and last names should only contain letters from the English alphabet. No numbers, no punctuation, no empty spaces. First and last names cannot be empty.
    - If this rule is violated, build_email should return "-1". (string)
- Year of entry should be a valid year: no letters, no punctuations. Also, year of entry must be 2001 or later.
    - If this rule is violated, build_email should return "-2". (string)
- If both rules are simultaneously violated, build_email should return "-1". (string)

Examples:

- firstname = “John”, lastname = “Doe”, entry_year = “2020”
  return value → “JDOE20@KU.EDU.TR”
- firstname = “Jane”, lastname = “Smith”, entry_year = “2005”
  return value → “JSMITH05@KU.EDU.TR”
- firstname = “John-Marcus”, lastname = “Doeington”, entry_year = “2010”
  return value → “-1”
- firstname = “James”, lastname = “Cool”, entry_year = “1998”
  return value → “-2”

Open EmailBuilder.py, implement your code inside the build_email function.

HINT: String operations in Lecture 9 will be really helpful ☺

Q2: Phone Number Converter - 25 pts

Many companies use telephone numbers like 555-GET-FOOD so the number is easier for their customers to remember. On a standard telephone, the alphabetic letters are mapped to numbers in the following fashion:

A, B, C: 2
D, E, F: 3
G, H, I: 4
J, K, L: 5
M, N, O: 6
P, Q, R, S: 7
T, U, V: 8
W, X, Y, Z: 9

Write a function phone_converter(phone_alpha) that:

- Takes as input phone_alpha: alphanumeric version of the phone number (such as “555-GET-FOOD” or “310-EAT-MEAT”)
- Returns a string in the format “XXX-XXX-XXXX” where each X is a digit corresponding to the letter in that position.

It is possible that the input string phone_alpha is not a valid phone number. To address these cases, you must check for erroneous inputs:

- Phone numbers starting with 0 are not allowed. In that case, your function should **return**: “-1” (string)

- Phone number format must be: 3 digits/characters, then a dash, then 3 digits/characters, then a dash, then 4 digits/characters. If the phone number does not follow this format or it is too long or too short, your function should **return**: “-2” (string).

Your function should work for: “555-GeT-fooD” the same way it would work for: “555-GET-FOOD”.

Examples:

- When phone_alpha = “555-GET-FOOD”, return value is: “555-438-3663”.
- When phone_alpha = “310-EAT-MeaT”, return value is: “310-328-6328”.
- When phone_alpha = “EAT-123-MEAT”, return value is: “328-123-6328”.
- When phone_alpha = “010-123-9876”, return value is: “-1”.
- When phone_alpha = “310-EAT-CHICKEN”, return value is: “-2”.
- When phone_alpha = “310-EATMEAT”, return value is: “-2”.
- When phone_alpha = “310-123456”, return value is: “-2”.
- When phone_alpha = “310-123-456”, return value is: “-2”.

Open PhoneConverter.py, implement your code inside the phone_converter() function. You can add your own functions as well.

Q3: Calculate Course Grade - 25 pts

Write a function that calculates the final aggregate grade for a course:
calculate_grade(lab_scores, hw_scores, mt_scores, final_exam_score)

The input parameters to the calculate_grade function are:

- lab_scores is a list containing the student’s lab scores.
  - Example: [100, 85.5, …, 90]
- hw_scores is a list containing the student’s homework scores.
  - Example: [83.5, 88, …, 100]
- mt_scores is a list containing the student’s midterm scores.
  - Example: [92.0, 85.5, …, 87]
- final_exam_score is a single value containing the student’s score in the final exam.
  - Example: 90.0

The return value of calculate_grade should be a single float that corresponds to the student’s final grade in this course, which is computed as follows.

- Lowest 2 out of 6 lab grades are dropped, then the average lab participation grade is calculated. This is worth 10% of the final grade.
- Lowest 2 out of 6 homework grades are dropped, then the average homework grade is calculated. This is worth 20% of the final grade.
- Lowest 2 out of 6 midterm grades are dropped, then the average midterm grade is calculated. This is worth 40% of the final grade.

- The final exam is worth 30% of the final grade.

You must check for the following erroneous inputs:

- lab_scores, hw_scores, mt_scores must contain exactly 6 values. If this rule is violated, then calculate_grade should return -1 (integer).
- All lab, homework, midterm, and final exam scores must be non-negative. If this rule is violated, then calculate_grade should return -2 (integer).
- If both rules are simultaneously violated, calculate_grade should return -1 (integer).

Open CalculateGrade.py, implement your code inside the function: calculate_grade.

Examples:

labs = [100, 85.5, 90, 90, 100, 78]
hws = [83.75, 82.0, 100, 100, 100, 100]
mts = [62, 91, 93, 59.0, 98.5]
final_score = 78.0
calculate_grade will return -1 because mts list includes 5 values.

labs = [100, 85.5, -5, 90, 100, 78]
hws = [83.75, 82.0, 100, 100, 100, 100]
mts = [62, 91, 93, 59.0, 98.5, 80]
final_score = 78.0
calculate_grade will return -2 because labs list includes a negative value.

labs = [100, 85.5, 90, 90, 100, 78]
hws = [83.75, 82.0, 100, 100, 100, 100]
mts = [62, 91, 93, 59.0, 98.5, 80]
final_score = 78.0
calculate_grade will return 89.15

Q4: Tic Tac Toe - 25 pts

Consider an NxN board for the Tic-Tac-Toe game, where N ≥ 3. You can simulate this board using a two-dimensional list (list of list). The game has two players: 'x' and 'o'. Each position on the board contains either:

- the character 'x'
- the character 'o'
- or the character '.' (dot) to denote that position is currently empty

Write a function tic_tac_toe(board) that takes as input an NxN Tic-Tac-Toe board and determines whether the game has been won by any of the players, or the game is still ongoing.

Player 'x' has won the game if:

- A row is entirely filled with 'x' values

- A column is entirely filled with ‘x’ values
- One of the 2 diagonals in the Tic-Tac-Toe board is entirely filled with ‘x’ values.

The same winning conditions apply to player ‘o’ (just swap ‘x’ by ‘o’ above).

If neither player has won, then the game is tied or it is still ongoing.

Your function tic_tac_toe should return the character ‘x’ if x has won, the character ‘o’ if o has won, or the character ‘.’ if the game has not been won by either player (tie or ongoing).

Important:

- Your function should work for any N ≥ 3. You can find N inside the tic_tac_toe function, from the input parameter board.
- You do not have to check for erroneous inputs. You can assume the input board is valid.
- You can assume that on a given board, there cannot be a situation where players ‘x’ and ‘o’ simultaneously satisfy the winning conditions.

Open TicTacToe.py, implement your code inside the function: tic_tac_toe.

Examples:

```
board = [['.',  'x',  '.',  'o'],
         ['.',  'x',  'o',  '.'],
         ['.',  'x',  '.',  '.'],
         ['o',  'o',  'x',  'o']]
```
tic_tac_toe will return ‘.’

```
board = [['.',  'x',  '.',  'o'],
         ['o',  'x',  'o',  '.'],
         ['.',  'x',  '.',  '.'],
         ['o',  'x',  'x',  'o']]
```
tic_tac_toe will return ‘x’

```
board = [['.',  'x',  'o'],
         ['x',  'o',  'x'],
         ['o',  '.',  '.']]
```
tic_tac_toe will return ‘o’

```
board = [['.',  'o',  'o'],
         ['x',  'x',  'x'],
         ['o',  '.',  '.']]
```
tic_tac_toe will return ‘x’

Submission and Grading

Solve each question in its own file. **DO NOT CHANGE THE NAMES OF THE FILES. DO NOT CHANGE THE HEADERS OF THE GIVEN FUNCTIONS (FUNCTION NAMES, FUNCTION PARAMETERS). DO NOT USE TURKISH CHARACTERS.** If you want, you can add helper functions.

When you are finished, compress your Hw2 folder containing all of your answers. The result should be a SINGLE compressed file (file extension must be .zip, or .rar). Upload this compressed file to Blackboard.

Follow instructions, print messages, input-output formats closely. **Your code may be graded by an autograder**, which means any inconsistency will be automatically penalized.

After you submit your Hw2, download it from Blackboard to make sure it is not corrupted and it has the latest version of your code. You are only going to be graded based on your Blackboard submission. **We will not accept homework via e-mail or other means**.

Happy coding! ☺