PGR107 - Python Programming

Kristiania University College

Final Exam

Academic contact during examination: Hadi Zahmatkesh, hadi.zahmatkesh@oslomet.no, +47 93255316

Technical contact during examination: eksamen@kristiania.no

Exam type: Written home examination in groups (1-5 students)

Support materials: All support materials are allowed

Plagiarism control: We expect your own independent work. Any copies from the internet or other groups will give you an automatic **FAIL** for every involved party (the giver and the taker).

Grading scale: Norwegian grading system using the graded scale A - F where A is the best grade, E is the lowest pass grade and F is fail.

- **A**: Excellent and comprehensive understanding of the topics
- **B**: Very good understanding of the topics
- **C**: Good understanding of the topics
- **D**: Satisfactory understanding of the topics but with significant shortcoming
- **E**: Meets the minimum understanding of the topics
- **F**: Fails to meet the minimum academic criteria

Learning outcomes (knowledge & skills):

The student

- understands problem solving using programming.

- understands the principles of object-oriented programming.

- has overall knowledge of general properties of python programming language such as program flow, loops, and choices.

- has knowledge of Python programming using data structures, functions, classes, objects, and modules.

Submit complete Python codes (.py files) for Questions 1-4.

Question 1 (25 points)

In this question, you will develop a **Word Guessing Game** using Python. The game involves guessing letters (A-Z) to form a word. The word is randomly chosen from a text file containing multiple words. Your program should read the file and select a word at random.

If the player guesses a correct letter that is part of the word, it should appear in its correct position(s). The player can continue guessing letters until they either complete the word or reach the maximum number of allowed guesses.

The maximum number of wrong guesses is equal to the length of the word. For example, if the word has 8 characters, the player can make up to 8 incorrect guesses before the game is over.

Here is a sample run:

```
The word you need to guess has '4' characters.

You have now 4 guesses.

_ _ _ _

Guess a character: a

Sorry that letter is not in the word.

_ _ _ _

you have 3 guess(es) left.

-------------------------

Guess a character: o

_ _ o _

you have 3 guess(es) left.

-------------------------

Guess a character: s

s _ o _

you have 3 guess(es) left.

-------------------------

Guess a character: n

s n o _

you have 3 guess(es) left.

-------------------------


Guess a character: w

You found the word --> "snow"
Congratulations! you won
```

<u>Here is another sample run:</u>

```
The word you need to guess has '4' characters.

You have now 4 guesses.

_ _ _ _

Guess a character: a

Sorry that letter is not in the word.

_ _ _ _

you have 3 guess(es) left.

-------------------------

Guess a character: b

Sorry that letter is not in the word.

_ _ _ _

you have 2 guess(es) left.

-------------------------

Guess a character: c

Sorry that letter is not in the word.

_ _ _ _

you have 1 guess(es) left.

-------------------------

Guess a character: d

Sorry that letter is not in the word.

_ _ _ _

you have 0 guess(es) left.

-------------------------

Sorry! you lost.
The word is --> "tree"
```

Question 2 (25 points)

In this question, you will implement a **Library Management System** in Python. Your implementation should include two main classes:

1. **Library** class – Represents the entity responsible for managing books.
2. **Book** class – Represents an individual book in the library.

Requirements:

Book class

- **Attributes:**
  - title – The title of the book.
  - author – The author of the book.
  - num_pages – The number of pages in the book.
- **Constructor (__init__)**
  - The initial values for all attributes should be provided when creating a book instance.

Library class

- **Attributes:**
  - books – A list that stores all books in the library.
- **Methods:**
  - add_book(book) – Adds a book to the library.
  - remove_book(title) – Removes a book from the library based on its title.
  - check_in(title) – Allows library visitors to return a book.
  - check_out(title) – Allows library visitors to borrow a book.

Write a Python program that implements the above classes. You may add additional methods and/or attributes if necessary to ensure the system functions correctly.

Write a test script to verify the functionality of your implementation. Your test should:

- Create a Library instance.
- Add multiple books to the library.
- Attempt to check out a book.
- Attempt to return a book.
- Remove a book from the library.
- Print the list of books to verify the results.

Question 3 (25 points)

In this question, you need to implement a **Bank Account Management System** in Python.

Your program should include the following classes, along with their respective properties (or attributes) and methods:

Menu class:

- **Property**: a list containing of all the menu options
- **Methods**:
  - __init__() - Initializes the list of menu options.
  - add_option() - Adds an option to the menu.
  - get_input() - Displays the menu and gets the user's choice.

BankAccount class:

- **Property**:
  - balance - Represents the account balance.
- **Methods**:
  - __init__() - Initializes the balance.
  - deposit() - Deposits money into the account.
  - withdraw() - Withdraws money from the account.
  - add_interest() - Adds interest to the current balance.
  - get_balance() - Returns the current balance.

Note: Your menu should appear as follows:

1 Open a new account

2 Deposit money into your account

3 Withdraw money from your account

4 Add interests to your account

5 Get the current balance of your account

6 Quit

Question 4 (25 points)

A string is considered a **palindrome** if it reads the same forwards and backwards (e.g., "anna", "civic", "level", and "hannah"). Write a Python **function** that checks whether a string entered by the user is a palindrome. Your program should:

- Prompt the user to input a string.
- Determine if the entered string is a palindrome.
- Display the result with a clear and meaningful output message indicating whether the string is a palindrome or not.

Make sure to handle the user input appropriately and consider edge cases (e.g., empty string, case sensitivity, spaces and punctuations, etc.) in your implementation.