LAB 2 – Writing a Complete Script in Python

Marks = 2

Submission =

- **For this exercise, you must work as per your lab group formed on BrightSpace. Parts 1 must be completed and checked in the class.**
- **After completing the 2 parts, submit your files (python files for parts 1 and 2) on Brightspace before the due date. Only one group member should submit the files. Only upload ".py" or ".txt" files.**

Part 1 – Working with date and time types in Python

For this exercise, carefully read about the date and time types in Python from the this documentation https://docs.python.org/3/library/datetime.html then do the following tasks:

1. Create a basic 'timedelta' and print it (expected result for example: 365 days, 5:01:00).
2. Print today's date and current time (Ex. Today is: 2023-01-12 00:23:50.465243).
3. Print today's date and time two years from now (Ex. Two years from now it will be: 2025-01-11 00:33:09.411866).
4. Create and print a timedelta with two arguments. E.g., In 2 weeks and 3 days it will be: 2023-01-29 00:33:09.411866. In this example, the output is a timedelta that generates its result based on two arguments 'weeks and days'. The result is also randomly generated based on current date and time. Thus, your code should also perform something in a similar way.
5. Calculate the date three weeks ago and print it like a string. Ex. Three weeks ago it was Thursday December 22, 2022.
6. Here you need to find the number of days till next Christmas. Your code should first check if Christmas has already gone for this year. If this is the case, then you should replace this year's Christmas date with next year's date. In the end, your code should print the number of days left till the next Christmas. Ex. 347 days left till next Christmas.

Part 2 – Create a Python program to manage a simple address book. The program should have the following features:

Read and write data to a text file named "address_book.txt". The data in the file should be structured as follows: Each line in the file contains a person's name followed by their corresponding email address, separated by a comma (e.g., "John Doe,johndoe@email.com"). When the program starts, it should load the data from the file into a dictionary where the names are the keys and the email addresses are the values.
The program should provide a menu with the following options:

- Add a new contact (name and email) by taking input from the user.
- Search for a contact by name and display their email address.
- List all contacts and their email addresses.
- Quit the program.

After each operation (add, search, list), the updated address book should be written back to the file. The program should keep running until the user chooses to quit.

Requirements:

You should use a text file to store the address book data.

Implement functions for each of the menu options to keep the code organized.

Handle errors gracefully, such as when a contact is not found during a search.

Use appropriate data structures (strings, lists and dictionaries) to store and manipulate the data.

Make sure to write the updated address book data to the file after each operation that modifies it.

Sample output:

```
Address Book Program
1. Add a new contact
2. Search for a contact
3. List all contacts
4. Quit

Enter your choice: 1
Enter name: Alice Smith
Enter email: alice@email.com
Contact added successfully!

Enter your choice: 3
Contacts:
John Doe - johndoe@email.com
```

```
Alice Smith - alice@email.com

Enter your choice: 2
Enter name to search: John Doe
Email for John Doe: johndoe@email.com

Enter your choice: 4
Goodbye!
```