***BE 1600***

***Introduction to***

***Programming and Computation***

***Python***

**Project 02**

40 points

**Due 12/04/2023 (11:45 A.M.)**

**Objectives**

* To use list of lists (2D Array)
* To use nested loops
* To use Python collections to store associative data
* To use text files to store large data sets
* To analyze real data

**Requirements:**

* Using Python IDE software, write a program for each of the problems. Includes informative comments.
* Test the code for each problem and verify that the program works.

**Restrictions:**

You must work individually. Use only material from class or from the textbook (chapters 1-9). All code must be the work of the individual. Do not share your code or copy from external resources.

**Submission**

Convert your python files to text files and submit two .txt files to Canvas by the due date

*Solution for this assignment will not be posted on Canvas; however, the solution of any of the assignment problems can be discussed in the class upon request of a student.*

All assignments must be submitted by the Canvas. **No email or hard copy** is accepted. You must follow the following format:

1. For non-programming questions, use a word file to type your answers. Don’t use the text box on the Canvas to answer the questions or to write comments, we will not read it. State your answer clearly.
2. For programming questions, include only the source file of each programming problem.
3. Submit your files to the Canvas. You must submit your files on time; otherwise, you will receive zero.
4. Use “Add Another File” feature on Canvas to upload each additional file; do not upload the files as a compressed folder.
5. You can upload your files as many times as you like. Only the last attempt counts because the last files you uploaded are the only files your instructor will see.
6. There will be several modules on the Canvas. You need to upload your files using the correct module on the Canvas.
7. Name each file: *Assignment (assignment number)* for the word file [e.g. Assignment 02] and *Assignment (assignment number) \_ (Question number)* for each programming question [e.g. Assignment 02\_Q03].
8. To upload your file(s):

* In Course Navigation, click the ASSIGNMENTS module.
* Click the title of the assignment.
* Click the **Submit** Assignment button.
* Add **File**. ...
* Add Another **File**. ...
* **Submit** Assignment. ...
* View **Submission**.

*It is your responsibility to make sure that each file is uploaded correctly. If you uploaded a wrong file, you receive zero; files will not be accepted after due date even if you have a prove that the file is created before the due date.*

***Make sure you review the Cheating & Plagiarism policy on Canvas.***

**Problem 01 (Memory Game) (20 points)**

Children often play a memory game in which a deck of cards containing matching pairs is used. The cards are shuffled and placed face down on a table. The players then take turns and select two cards at a time. If both cards match, they are left face up; otherwise, the cards are placed face down at the same positions. Once the players see the selected pair of cards and if the cards do not match, then they can memorize the cards and use their memory to select the next pair of cards.

The game continues until all the cards are face up. Write a program to play the memory game. **Use a two-dimensional list** of 4 rows and 4 columns for a deck of 16 cards with 8 matching pairs. You can use numbers 1 to 8 to mark the cards. Use random number generators to randomly store the pairs in the array.

Your program should consist of at least the following functions:

1. A function to fill the board randomly.
2. A function to play the game
3. A function to get the card location.
4. A function to show the Board.
5. A main function

**Grading Criteria:**

4 points for each function (a – e)

2 points will be deducted for each of the followings if not implemented:

* Generating the cards randomly.
* Checking for card invalid position.
* Handling card already faced up case.
* Terminating the game when all cards are face up.
* Comments for each function

***Problem 01 Sample Output***

|  |
| --- |
| \* \* \* \*  \* \* \* \*  \* \* \* \*  \* \* \* \*  Enter the row (1 to 4) and col (1 to 4) position of the pair:**2 2**  \* \* \* \*  \* 8 \* \*  \* \* \* \*  \* \* \* \*  Enter the row (1 to 4) and col (1 to 4) position of the pair:**3 4**  \* \* \* \*  \* 8 \* \*  \* \* \* 1  \* \* \* \*  Pair do not match. Select again!  \* \* \* \*  \* \* \* \*  \* \* \* \*  \* \* \* \*  Enter the row (1 to 4) and col (1 to 4) position of the pair:**1 2**  \* 1 \* \*  \* \* \* \*  \* \* \* \*  \* \* \* \*  Enter the row (1 to 4) and col (1 to 4) position of the pair:**3 4**  \* 1 \* \*  \* \* \* \*  \* \* \* 1  \* \* \* \*  Pair match  \* 1 \* \*  \* \* \* \*  \* \* \* 1  \* \* \* \*  Enter the row (1 to 4) and col (1 to 4) position of the pair:**6 2**  Invalid position.  Enter the row (1 to 4) and col (1 to 4) position of the pair:**1 2**  Card at this position already faced up. Select position again.  Enter the row (1 to 4) and col (1 to 4) position of the pair:**1 1**  5 1 \* \*  \* \* \* \*  \* \* \* 1  \* \* \* \*  Enter the row (1 to 4) and col (1 to 4) position of the pair:**1 3**  5 1 8 \*  \* \* \* \*  \* \* \* 1  \* \* \* \*  Pair do not match. Select again!  \* 1 \* \*  \* \* \* \*  \* \* \* 1  \* \* \* \*  Enter the row (1 to 4) and col (1 to 4) position of the pair:**1 3**  \* 1 8 \*  \* \* \* \*  \* \* \* 1  \* \* \* \*  Enter the row (1 to 4) and col (1 to 4) position of the pair:**2 2**  \* 1 8 \*  \* 8 \* \*  \* \* \* 1  \* \* \* \*  Pair match  \* 1 8 \*  \* 8 \* \*  \* \* \* 1  \* \* \* \*  Enter the row (1 to 4) and col (1 to 4) position of the pair:**1 4**  \* 1 8 4  \* 8 \* \*  \* \* \* 1  \* \* \* \*  Enter the row (1 to 4) and col (1 to 4) position of the pair:**2 4**  \* 1 8 4  \* 8 \* 2  \* \* \* 1  \* \* \* \*  Pair do not match. Select again!  \* 1 8 \*  \* 8 \* \*  \* \* \* 1  \* \* \* \*  Enter the row (1 to 4) and col (1 to 4) position of the pair:**2 1**  \* 1 8 \*  6 8 \* \*  \* \* \* 1  \* \* \* \*  Enter the row (1 to 4) and col (1 to 4) position of the pair:**3 1**  \* 1 8 \*  6 8 \* \*  7 \* \* 1  \* \* \* \*  Pair do not match. Select again!  \* 1 8 \*  \* 8 \* \*  \* \* \* 1  \* \* \* \*  Enter the row (1 to 4) and col (1 to 4) position of the pair:**4 4**  \* 1 8 \*  \* 8 \* \*  \* \* \* 1  \* \* \* 2  Enter the row (1 to 4) and col (1 to 4) position of the pair:**2 4**  \* 1 8 \*  \* 8 \* 2  \* \* \* 1  \* \* \* 2  Pair match  \* 1 8 \*  \* 8 \* 2  \* \* \* 1  \* \* \* 2  Enter the row (1 to 4) and col (1 to 4) position of the pair:**2 3**  \* 1 8 \*  \* 8 3 2  \* \* \* 1  \* \* \* 2  Enter the row (1 to 4) and col (1 to 4) position of the pair:**2 4**  Card at this position already faced up. Select position again.  Enter the row (1 to 4) and col (1 to 4) position of the pair:**3 3**  \* 1 8 \*  \* 8 3 2  \* \* 4 1  \* \* \* 2  Pair do not match. Select again!  \* 1 8 \*  \* 8 \* 2  \* \* \* 1  \* \* \* 2  Enter the row (1 to 4) and col (1 to 4) position of the pair:**4 1**  \* 1 8 \*  \* 8 \* 2  \* \* \* 1  7 \* \* 2  Enter the row (1 to 4) and col (1 to 4) position of the pair**:3 1**  \* 1 8 \*  \* 8 \* 2  7 \* \* 1  7 \* \* 2  Pair match  \* 1 8 \*  \* 8 \* 2  7 \* \* 1  7 \* \* 2  Enter the row (1 to 4) and col (1 to 4) position of the pair:**3 2**  \* 1 8 \*  \* 8 \* 2  7 5 \* 1  7 \* \* 2  Enter the row (1 to 4) and col (1 to 4) position of the pair:**4 2**  \* 1 8 \*  \* 8 \* 2  7 5 \* 1  7 3 \* 2  Pair do not match. Select again!  \* 1 8 \*  \* 8 \* 2  7 \* \* 1  7 \* \* 2  Enter the row (1 to 4) and col (1 to 4) position of the pair:**2 3**  \* 1 8 \*  \* 8 3 2  7 \* \* 1  7 \* \* 2  Enter the row (1 to 4) and col (1 to 4) position of the pair:**4 2**  \* 1 8 \*  \* 8 3 2  7 \* \* 1  7 3 \* 2  Pair match  \* 1 8 \*  \* 8 3 2  7 \* \* 1  7 3 \* 2  Enter the row (1 to 4) and col (1 to 4) position of the pair:**4 3**  \* 1 8 \*  \* 8 3 2  7 \* \* 1  7 3 6 2  Enter the row (1 to 4) and col (1 to 4) position of the pair:**2 1**  \* 1 8 \*  6 8 3 2  7 \* \* 1  7 3 6 2  Pair match  \* 1 8 \*  6 8 3 2  7 \* \* 1  7 3 6 2  Enter the row (1 to 4) and col (1 to 4) position of the pair:**3 1**  Card at this position already faced up. Select position again.  Enter the row (1 to 4) and col (1 to 4) position of the pair:**4 1**  Card at this position already faced up. Select position again.  Enter the row (1 to 4) and col (1 to 4) position of the pair:**1 1**  5 1 8 \*  6 8 3 2  7 \* \* 1  7 3 6 2  Enter the row (1 to 4) and col (1 to 4) position of the pair:**3 2**  5 1 8 \*  6 8 3 2  7 5 \* 1  7 3 6 2  Pair match  5 1 8 \*  6 8 3 2  7 5 \* 1  7 3 6 2  Enter the row (1 to 4) and col (1 to 4) position of the pair:**1 4**  5 1 8 4  6 8 3 2  7 5 \* 1  7 3 6 2  Enter the row (1 to 4) and col (1 to 4) position of the pair:**3 3**  5 1 8 4  6 8 3 2  7 5 4 1  7 3 6 2  Pair match  5 1 8 4  6 8 3 2  7 5 4 1  7 3 6 2  Enter any key to continue . . . |

**Problem 02 (Text Analysis) (20 points)**

Write a program that reads text from two text files and write to an output file:

* List of all the unique words contained in the first file.
* List of all the unique words contained in the second file.
* List of all the unique words contained in both files.
* List of the words that appear in both files.
* List of the words that appear in the first file but not the second file.
* List of the words that appear in the second file but not the first file.
* List of the words that appear in either the first or second file but not both files.
* Frequency table of words counts in each file

The program should have the following functions:

1. A function to write data from a list to a file
2. A function that creates a list of unique words from a file and returns it
3. A function that creates a list of union of words in two files and returns it
4. A function that creates a list of common words in two files and returns it
5. A function that creates a list of words in one file but not in another file and returns it.
6. A function that prints in tabular format the count of each word in a file.
7. Main function to call all the above functions.
8. A function to read data from a file and returns a list of words in a file

*Make sure you remove punctuations at the end of each word. Do not Use set operations to perform the above analyses.*

**Grading Criteria:**

2 points for each function (a – f)

4 points for each function (g – h)

2 points will be deducted if punctuations are not removed.

10 points will be deducted if you use set operations.

***Here is sample run for input files, file1.txt and file2.txt:***

|  |  |
| --- | --- |
| **file1.txt**  one, two, three, four, five;  six seven:  eight!  nine ten eleven:  twelve?  one two three. | **File2.txt**  one, two, three four five, six  anotherseven!  anothereight?  nine ten eleven  twelve:  ten eleven twelve; |

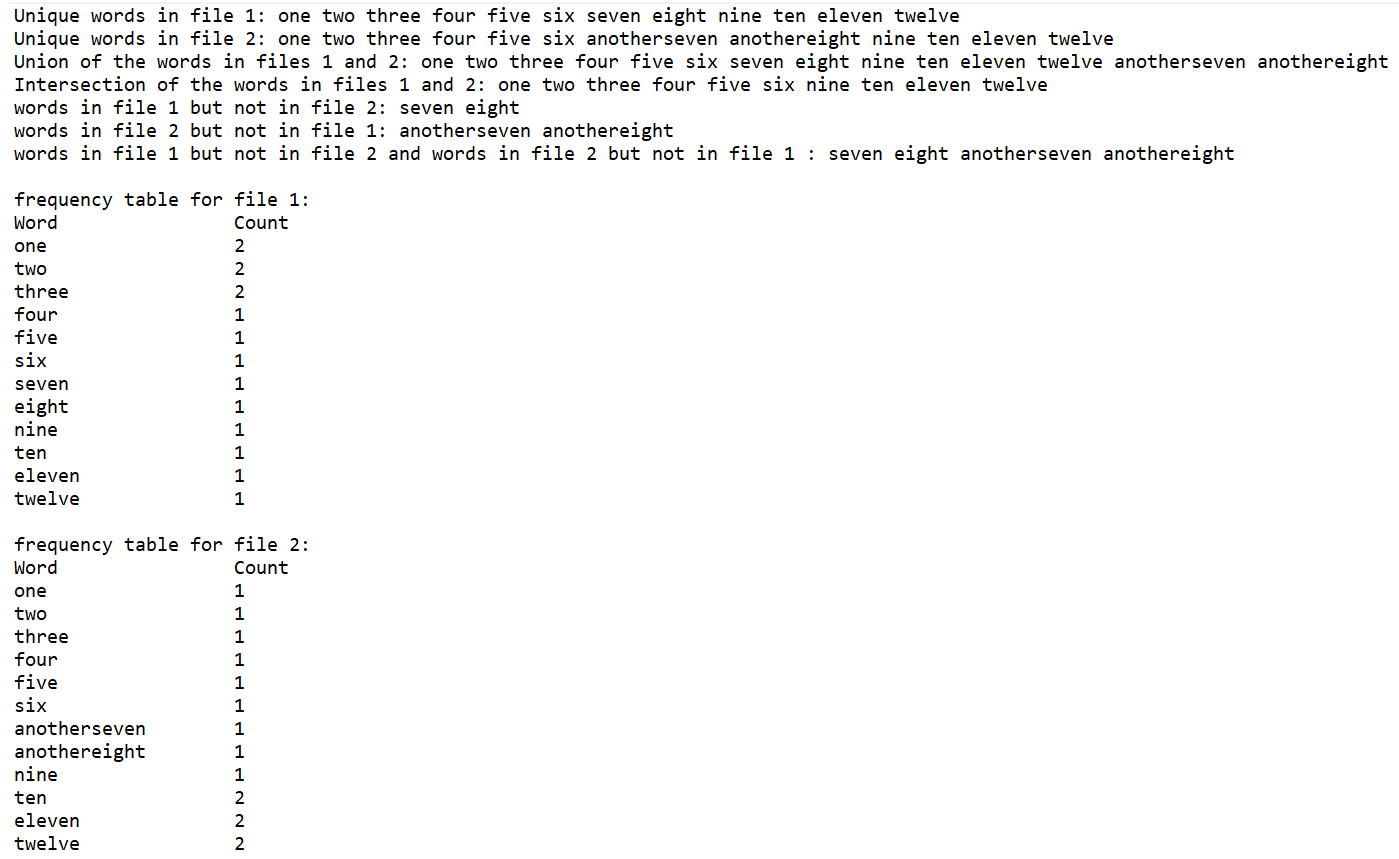
*Output in shell:*

Enter the name of the first input file: file1.txt

Enter the name of the second input file: file2.txt

data saved in fileAnalysis.txt

*Output in file:*



**Extra Credit (Name and Email Addresses) (25 points)**

Write a menu driven program that keeps names and email addresses in a dictionary as key-value pairs.

The program should display a menu that lets the user look up a person’s email address, add a new name and email address, change an existing email address, and delete an existing name and email address. The program should save the data stored in a dictionary to a file when the user exits the program. Each time the program starts, it should retrieve the data from the file and store it in a dictionary.

You must create and use at least 5 meaningful functions.

1. A function to display a menu
2. A function to look up a person’s email address
3. A function to add a new name and email address
4. A function to change an email address
5. A function to delete a name and email address.

Your program should check user input; for example, if the user wants to delete a name that does not exist in the dictionary, you should print something like: the name is not in the database. A sample one run is shown below.

**Grading Criteria:**

3 points for each function (a – e)

5 points for creating menu driven program.

5 points for loading the data at the beginning of the program from a file and save it at the end of the program to a file.

5 points will be deducted for not validating user input.

|  |
| --- |
| Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 2  Enter name: John  Enter email address: John@yahoo.com  That name already exists  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 2  Enter name: Jack  Enter email address: Jack@yahoo.com  Name and address have been added  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 1  Enter a name: Sam  The specified name was not found  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 1  Enter a name: Jack  Name: Jack  Email: Jack@yahoo.com  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 3  Enter name: John  Enter the new address: John@wayne.edu  Information updated  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 1  Enter a name: John  Name: John  Email: John@wayne.edu  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 4  Enter name: Sam  The specified name was not found  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 4  Enter name: Jack  Information deleted  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 1  Enter a name: Jack  The specified name was not found  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 5  Information saved  >>> |