**task**

LifeServe Blood Institute (LBI) are happy with your performance in the previous project. They have decided to engage you for developing a more featureful software that will help LBI staff carry out their day to day jobs. The proposed system will provide several features such as managing inventory records, storing donor details and attending to blood demand. In addition, the system will enforce the following medical rules:

* Once a blood bag is collected, it must be used within 30 days. After that period, the bag is discarded.
* For every donor, there must be a minimum gap of 120 days between donations. During the gap period, a donor is labelled as ineligible.

**What you are provided**

Download [this zip package](https://doms.csu.edu.au/csu/file/8a2be334-f327-4343-9d8e-fb8f18df545d/1/ITC558_A3_Data.zip) containing sample database files to be used by LBI system. The file donors.txt contains the records of donors currently registered in the system. Each line in the file contains donor data according to following format:

donor\_id,name,phone,email,blood\_group,last\_donation\_date

The date field is stored in standard ISO format (YYYY-MM-DD).

Secondly, the file bags.txt (and another sample oldbags.txt) contain records of blood bags currently in stock at LBI. Format of each bag record is:

bag\_id,blood\_group,date\_collected

In both files, records are sorted by ID numbers in ascending order.

**Preliminary steps**

Create a Python dictionary to store the [blood transfusion compatibility table](https://mytransfusion.com.au/sites/default/files/styles/content_section/public/Red%20cell%20compatibility.png). Dictionary should be set up such that you could quickly look up a patient’s blood group to retrieve a list of compatible blood types. For example, dict['B+'] would return ['O-', 'O+', 'B-', 'B+'].

Create a function load\_db(donor\_fname, stock\_fname) that takes in two file names as parameters, reads both files and stores their data in appropriate data structures (lists, tuples, dictionaries etc.). This function will be called only once in the beginning of the program.

Create a converse function save\_db(donor\_fname, stock\_fname) that takes the data currently in memory and dumps it into two files in the exact same format as described above. save\_db function would be called multiple times during the program execution whenever donor or inventory information is modified. Please keep the original files (from zip archive) intact – store the modified database files with new names bags-new.txt and donors-new.txt.

**Main Menu**

As your program starts, you should prompt the user for two database file names, load their data into memory (via load\_db), initialize the blood compatibility dictionary and then present the user with a main menu containing five options.

(1) Check inventory: With this option, the program will search for any bags older than 30 days, and if found, display their ID numbers so that staff can dispose them of. The database should be updated right away.

(2) Attend to blood demand: When the local hospital needs a supply of blood, they contact LBI to arrange it. With this option, your program should check what type of blood is currently needed at the hospital. The zip file provided to you contains a module hospital.py containing a single function check\_demand(). Call this function to find out the blood type required by hospital. [Let us pretend this module is communicating with hospital web servers.] The function returns a single string value containing the blood type. In some cases, the function may return an X value to indicate server communication errors. Your program will then check the inventory to see if a bag with matching or compatible blood type exists in the stock.

* If compatible bag(s) are found in stock, the program pick any one and display its ID number so that staff can prepare it for dispatch to hospital. The said bag should be removed from database.
* Otherwise, the next task is to check the database of donors and find a list of eligible donors with a compatible blood type (eligibility criteria as defined earlier). A list of names will be displayed along with their contact details so that staff can communicate with them.
* If no eligible donor exists in database, just display an informational message.

(3) Record new donation: This option would allow the staff to add a new bag to the database. It will first ask for a valid donor ID and check their eligibility to donate. If eligible, a new bag will be added with current date (today, the time of execution) and a new auto generated ID number (increment the ID of the last bag). The respective donor’s last donation date will also be updated. Database files should be saved once addition is confirmed.

(4) Stock visual report: Allow the user to see the distribution of in-stock blood bags in the form of a pie chart. Label the pies with blood types and the current number of bags in stock. Do not show the blood types with zero stock.

(5) Exit the program

To clearly understand the program requirements, [study these sample runs](https://doms.csu.edu.au/csu/file/8a2be334-f327-4343-9d8e-fb8f18df545d/1/ITC558_A3_sample_run.html).

**Handling invalid inputs**

Your program is supposed to handle, at a minimum, the following invalid inputs or exceptional situations. Some of these (but not all) are demonstrated in the sample runs.

* Incorrect database file names
* File reading/writing errors
* Invalid options for main meu
* Invalid donor IDs (in menu option 4)

**Constraints**

In addition to hospital (and your own modules), you can only import the following library modules: datetime, matplotlib, sys

**Sample Codes**

(1) Working with dates

from datetime import date

some\_date = date.fromisoformat('2019-12-04')

today = date.today()

days\_diff = (today - some\_date).days # to calculate difference in number of days

print(days\_diff)

print(some\_date.isoformat()) # to represent a date object in ISO format

(2) Creating the pie chart

Use matplotlib library. See installation and usage instructions in Chapter 7 of the textbook.

---

Your assignment should consist of the following tasks.

**Task 1**

Draw  *high-level* flowcharts that represent the algorithms of **menu option 2** and **menu option 3**. High-level flowchart means it does not need to map exactly to your code or function calls. The goal of such a flowchart is to illustrate the logic of your algorithm without worrying about code intricacies.

You can draw the flowcharts with a pen/pencil on a piece of paper and scan it for submission, as long as the handwriting is clear and legible. However, it is strongly recommended to draw flowcharts using a drawing software.

**Task 2**

Select **six** sets of test data that will demonstrate the 'normal' operation of your program; that is, test data that will demonstrate what happens when a valid input is entered. Select **four** sets of test data that will demonstrate the 'abnormal' operation of your program.

A lot of input to this program is not via keyboard, but through text files. Therefore, to create test case, you should create **additional database files** with modified data. Include all of those files in your submission so that marker can verify your test cases.

Set out the test cases in a tabular form as follows. It is important that the output listings (i.e., screenshots) are not edited in any way.

|  | | | | |
| --- | --- | --- | --- | --- |
| **Test Data Table** | | | | |
| **Test data type** | **Test data** | **The reason it was selected** | **The output expected due to the use of the test data** | **The screenshot of actual output when the test data is used** |
| Normal |  |  |  |  |
| Normal |  |  |  |  |
| Abnormal |  |  |  |  |
| Abnormal |  |  |  |  |

**Task 3**

Implement the required Python program. Comment on your code as necessary to explain it clearly.  Run your program using the test data you have selected and complete the final column of test data table above.

**Your submission will consist of:**

1. Your algorithms through flowcharts.
2. The table recording your chosen test data and results (it should be a PDF file)
3. Source code for your Python implementation
4. Alternate database files used for test cases. Name them donors1.txt, donors2.txt, bags1.txt, bags2.txt and so on.

These files should be compressed into a single ZIP before uploading in Turnitin.  
  
It is critically important that your test runs are unmodified outputs from your program, and that these results should be reproducible by the marker running your saved .py python program.

SAMPLE RUNS

# Sample Run 1

<<< LifeServe Blood Institute >>>

Loading database...

Enter the database file names without .txt extension

or just press Enter to accept defaults

Donors database (donors): **[Enter]**

Stock inventory database (bags): **oldbags**

Database loaded successfully

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

Main Menu

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

(1) Check inventory

(2) Attend to blood demand

(3) Record new donation

(4) Stock visual report

(5) Exit

Enter your choice: **1**

Following bags are out of their use-by date

3574

3575

3576

3590

Please dispose them of. Press any key when done...

Updated database files saved to disk.

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

Main Menu

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

(1) Check inventory

(2) Attend to blood demand

(3) Record new donation

(4) Stock visual report

(5) Exit

Enter your choice: **5**

Have a good day.

# Sample Run 2

<<< LifeServe Blood Institute >>>

Loading database...

Enter the database file names without .txt extension,

or just press Enter to accept defaults.

Donors database (donors): **[Enter]**

Stock inventory database (bags): **[Enter]**

Database loaded successfully.

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

Main Menu

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

(1) Check inventory

(2) Attend to blood demand

(3) Record new donation

(4) Stock visual report

(5) Exit

Enter your choice: **2**

Currently B+ blood is required.

Checking the stock inventory...

Following bag should be supplied

ID: 3590 (B-)

Press any key once it is packed for dispatch...

Inventory records updated.

Updated database files saved to disk.

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

Main Menu

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

(1) Check inventory

(2) Attend to blood demand

(3) Record new donation

(4) Stock visual report

(5) Exit

Enter your choice: **2**

Could not connect to hospital web server.

Please try again after some time.

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

Main Menu

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

(1) Check inventory

(2) Attend to blood demand

(3) Record new donation

(4) Stock visual report

(5) Exit

Enter your choice: **2**

Currently B- blood is required.

Checking the stock inventory...

We can not meet the requirement. Checking the donor database...

Following donors match the requirements. Please contact them for new donation.

• Jorgen Krysia, 0462 316 524, jkrysia@defence.gov.au

• Gafar Farrukh, 0431 252 844, gfarrukh@csu.edu.au

• Trygve Imke, 0443 023 775, trygve1995@gmail.com

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

Main Menu

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

(1) Check inventory

(2) Attend to blood demand

(3) Record new donation

(4) Stock visual report

(5) Exit

Enter your choice: **3**

Enter the donor's unique ID: 5200

That ID does not exist in the database.

To register a new donor, please contact the system administrator.

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

Main Menu

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

(1) Check inventory

(2) Attend to blood demand

(3) Record new donation

(4) Stock visual report

(5) Exit

Enter your choice: **3**

Enter the donor's unique ID: **1251**

Sorry, this donor is not eligible for donation.

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

Main Menu

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

(1) Check inventory

(2) Attend to blood demand

(3) Record new donation

(4) Stock visual report

(5) Exit

Enter your choice: **3**

Enter the donor's unique ID: **1256**

Recording a new donation with following details:

From: Akanksha Adelardo

Group: A+

Date: 2021-02-03 (today)

Bag ID: 3590

Please confirm (y/n): **n**

Cancelled.

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

Main Menu

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

(1) Check inventory

(2) Attend to blood demand

(3) Record new donation

(4) Stock visual report

(5) Exit

Enter your choice: **3**

Enter the donor's unique ID: **1252**

Recording a new donation with following details:

From: Jorgen Krysia

Group: B-

Date: 2021-02-03 (today)

Bag ID: 3590

Please confirm (y/n): **y**

Done. Donor's last donation date also updated to 2021-02-03

Updated database files saved to disk.

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

Main Menu

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

(1) Check inventory

(2) Attend to blood demand

(3) Record new donation

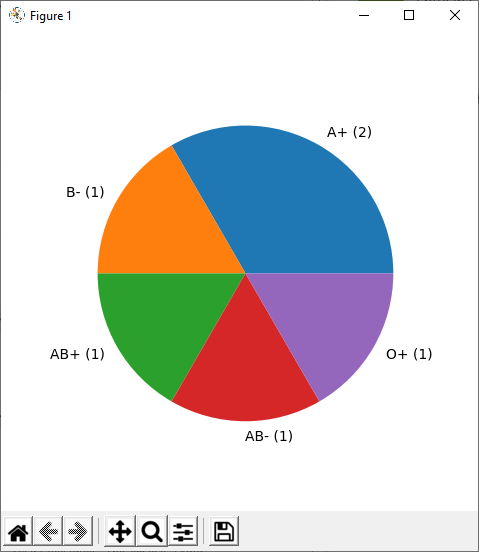
(4) Stock visual report

(5) Exit

Enter your choice: **4**

Pie chart opens in a new window...

Close the chart window to continue



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

Main Menu

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

(1) Check inventory

(2) Attend to blood demand

(3) Record new donation

(4) Stock visual report

(5) Exit

Enter your choice: **5**

Have a good day.

[blood transfusion compatibility table](https://mytransfusion.com.au/sites/default/files/styles/content_section/public/Red%20cell%20compatibility.png)

