Operating system-1- Final project

Q1:

Databases are usually hard to install and maintain, so our company (with the top-notch engineers) is planning to release a new database engine into the market and the main focus is the portability and security. The first phase of the plan is to release a demo version of the database, fortunately, you are lucky because you have been selected to be in the engineering team for the demo version. And the deadline is almost on the line, so harry up and finish the requirements:

Please write a **key -> value database** using your bash scripting knowledge to provide very basics functionality as following:

1. Insert a new record (key -> value)
2. Search for a record by key
3. Delete a record by key
4. Update a record value by key

Also keep in your mind that the structure of the database file should match the following image:

```
version 1 os1
ITE1 : dGhpcyBpcyBhIHRlc3Q=
this8is8a8key : Zm9ydGggeWVhcg==
```

Here is the description of the structure (for each line):
Line 1 is the header: "version" + SPACE + "1" + SPACE + $DB_OWNER_NAME
Line 2 is a record: $KEY + SPACE + ":" + SPACE + $VALUE (encoded in base64)
Line 3 to INF: is also a record (same as line 2)
To ease the use of our database engine, the engineers had an amazing idea, which is building a terminal interface for our database demo version, our designer draw something cool check it out:

```
Q1 (first.os1db):
1) Add new record
2) Delete a record
3) Search for a record
4) Update a record
Select an option please: 1
Please enter the key: ITE1
Please enter the value: this is a test
Adding new key to the database (first.os1db)...
Done
Q1 (first.os1db):
1) Add new record
2) Delete a record
3) Search for a record
4) Update a record
```

I think our engineers is very smart and do not need any help to understand how would they build the terminal interface based on our designer draw.

Notes:

- We want our database demo to be lunched like that:

  ./db_demo.sh $DATABASENAME

  Where $DATABASENAME: is an existing file (relative or absolute path)
- We also want our bash database engine to support a flag for creating the database:

  ./db_demo.sh --create $DATABASENAME

  The --create flag: will create the database if the database does not exist

- Finally, **the most important** note is to split the database engine into 2 scripts:
  1. Core script: responsible for the core functionality (functions)
  2. Terminal UI script: The terminal UI script that import the core script to provide the functionality

Q2:

After the demo version, our customers are very impressed, and the engineers had a great vacation It's time for releasing new features! Our customers are sending emails to us non-stop to add backup and restore features so you have prioritize our goal for the new version and we have already assigned the engineers (I guess you know who!)

1. Please add a backup feature to our database that support multiple compression mode (zip, tar and gzip)
2. Please also add a restore feature to our database engine
3. Finally, schedule backups (with rotation) will be a nice touch to be included in the next release.

If you are waiting for the designer work here it is:

```
Q2 backup / restore (first.os1db):
1) Backup the database
2) Restore a database
3) Enable automatic backups
Please select your options: 1
Enter the backup name (with file extension .zip or .gz): first_backup1.zip
Using zip for backup (file has .zip extension)...
Backup done (/home/os1/first_backup1.zip)
Q2 backup / restore (first.os1db):
1) Backup the database
2) Restore a database
3) Enable automatic backup
Please select your options: 3
Please enter your backup schadlule (daily, weekly or monthly): daily
Number of max backup files to keep: 10
Adding (0 0 * * * /home/os1/q2.sh /home/os1/first.os1db --backup --outputdir
/opt/backups/ --max 10)
Done.
```

Notes:

- Please take a closer look at our designer work especially the line with the cronjob syntax (try to wake up your brain be drinking café)
- The manger wants the restoration process to launch the database in our first demo version script (after the restoration)

Q3:

We want to monitor our system as it is an essential task for system administrators to monitor system performance. we are going to write a bash script to monitor our system. The script will show three columns showing the percentages of Memory, Disk and CPU used on our system. The script will run for a certain amount of time every 5 seconds for an hour (It measures performance every five seconds for an hour only Starting from the time the script is executed)

Notes:

- The script will run automatically every day at 12 AM .
- The output will be saved to a text file (monitoring.txt) within the path (/opt/monitoring.txt)
- Print in the file (monitoring.txt) the date of each day the script will work.

Q4:

Our engineers are so tired from the database work…, so we have a small project needs a hand and you have been assigned to it.
Our company wants to build a local FTP Server to store some global files, so it is your job to install and configure the FTP Server on the company server.

1. Install and configure the FTP server
2. Back to bash scripting: We have an emergency situation and we need a script to upload a file or folder from our pc to the ftp server
3. Also, we want a small script to download a file or folder from the FTP server to our pc

Q5: Additional question for groups above 3.

looking for files and directories in linux can be a little tricky, as commands exist to help us find certain things, we need more features in the searching process, write a script in which you can as a elite engineer search for wanted files or dirs. The script should provide these features:
1- Take more than one file name or dir to search for it, hence multithreaded.
2- As similar file names exist in linux you have to seek to provide the best results to the user.
3- Show the first 3 columns from ls -l command (permissions, owner, group).
4- Users tend to search a lot for the same thing over and over, find a way to cache the recent results for fast access time.

Notes:
1-The number of students in each group :
Networks and operating systems 3 students only.
Software engineering and artificial intelligence 3 student for group. Any group above 3 (4 or 5 only) Added to them the fifth question.
2- Write one report for each group explaining how the script works for each question, with screenshots of the implementation and send a soft copy of the report to the Email:
operatingsystem.1.ite@gmail.com end of the day 20-11-2022
3- Interview date: 21-11-2022 with a hard copy of report.

Best wishes ^-^