(a) a general overview of your system with a small user guide

Project2 contains four different files, Phase1.py, Phase2.bash, Phase3.py and break.pl.

Phase1.py:
This file must be run in the command of “python3 Phase1.py inputfile”, where inputfile is the file that you need to input. After that, you will get four different files, “term.txt”, “pdates.txt”, “prices.txt” and “ads.txt”.

main():

1. GetFileName(): This function will get the system arguments from the command line, and return the input file name.
2. Then the main will open five different txt files for reading and writing.
3. The program will read the input file line by line.
4. It will subtract aid first and check if aid exists or not
5. If exists, then subtract title description date, category, location and price by using function substraction()
6. Then it would use function simplified twice for write title and description in output file “terms.txt”
7. It would also write date, rice and full record in “pdates.txt”, “prices.txt” and “ads.txt”
8. Finally, the program will close all the txt files which are opened before.

Phase2.bash:
This bash script will be run in the command of “bash Phase2.bash”. After that you will get the sorted files from Phase1 and four new index files named “ad.idx”, “te.idx”, “da.idx” and “pr.idx”.

There are eight different commands, four for sorting files using “sort -u -o” and the rest of the commands can input all four files into four different index. Firstly, the command will cat txt files and using break.pl to get rid of the sign and then use db_load to load them into the index files.

Phase3.py:
This file must be run in the command of “python3 Phase3.py inputfile”, where the inputfile must be the exact same file in Phase1. After that, you can see the home menu in the screen with option of retrieving data and quitting the program. If you wish to choose retrieving data, then you will be asked to input the conditions you want to search. After inputting the data, you will get the corresponding result.

In general speaking, there are nine major functions.

1. GetFilename: This will get the filename of the input file.
2. CheckFormat: This will check if “output = brief/full” is in the input conditions.
3. CheckPrice: This will check if “price” is in the conditions and return all matching aids.
4. CheckDate: This will check if “date” is in the conditions and return all matching aids.
5. CheckLocation: This will check if “location” is in the conditions and return all matching aids.

6. CheckCat: This will check if “cat” is in the conditions and return all matching aids.
7. CheckOthers: This will check the remaining conditions in te.idx and return all matching aids.
8. Combine: This will combine all aids and return the final result of aids.
9. DisplayResult: This will print out the result based on the output format.

(b) a description of your algorithm for efficiently evaluating queries

The program will get a dictionary of title, where the key is the aid and the corresponding data is the full title. Since we don’t think there is a way to get the full title from te.idx since some of the words are erased because of the short length. So we think this is the most efficient way of doing this.

For printing the full record, we use ad.idx by searching aid using cursor.set. So we believe this is the most efficient way of doing this.

For finding the price, we use the checkrange which was taught in the lab. We need to access pr.idx to do the range search. We mainly use the cursor.set_range to make sure the efficiency. We believe this is the most efficient way of doing this.

For find the date, it is almost the same as finding the rice, so we believe this is the most efficient way of doing this.

For finding the location and the category, we iterate the whole pr.idx/da.idx and compare the data of each key. We believe there is a better way to do this, but so far we haven’t come up a solution yet.

For finding others, there are two parts. For finding the exact match, we use the cursor.set on te.idx which we believe this is the most efficient way of doing this. Bu for finding the partial match, we iterate all terms in te.idx and do the comparison, we believe there is a better way to do this, but so far we haven’t come up a solution yet.

For dealing with multiple conditions, we combine all matching aids for each matching term by intersection. We believe there is a better way to do this, we can deal with the first condition and find other conditions based on the returned aids of the first condition, but we don’t have enough time to change.

(c) testing strategy, and (d) group work break-down strategy.

For Phase1 and Phase 2, we tested on 10.txt, 10k.txt and 20k.txt by comparing the posted files and our own files. We test each detailed conditions in 10.txt, and we tested some random conditions in 10k.txt and 20k.txt.

For this whole program, we met roughly 4-5 times(3-5 hours per time) in the lab to work together. So we wrote everything in this program together. There are some small functions which were wrote by each individual but there are too detailed to keep on track.