CSE 30 Fall 2022 Programming Assignment #7 - (Vers 1.0) Due Friday November 11 , 2022 @ 11:59PM

Remember: 'Start Early'

Grading (50 points total) What You Need To Do

You must complete all of the following items by the due date to get all points.
The last day to turn in is Monday Nov 14, 2022 at 11:59 PM.
There are **50 points available** for this PA that are distributed as follows:

- ☐ Up to 4 points for following the C style guidelines. https://docs.google.com/document/d/1jltOTaolKNfF7QTnCnJQ9KxgkcllDJ_rD86u7nHnHZ4/edit?usp=sharing
- ☐ Up to 6 points (at our discretion) if the following files compile without warnings. This has to be a meaningful attempt to write the code that meets the specifications of the program. Random C statements or empty files will not be awarded any points. **These are the only files that are part of PA6:**
  ```
  insertticket.c (3 points)
  delticket.c (3 points)
  ```
- ☐ Up to 25 points for passing the tests in the included test harness
- ☐ Up to 10 points for passing the gradescope tests (these will be run after the late deadline).
- ☐ Up to 5 points canvas checkpoint quiz (due Tuesday 11/8). No slip days or late submissions.

Assignment – Pointers and Data Structures - a Memory based Database

PA7 is the second part of a two part project where you will continue working on an in-memory parking ticket database. See PA6 writeup for a description of the two functions that need to be written for PA7.

Writing the two replacement routines for PA7

You will be developing this in the SAME github repository you coned for PA6, no additional files are needed.

1. **Write ONLY the following replacement functions for PA7 in any order you like.**
   ```
   insertticket()  in file: insticket.c
   delticket()     in file: delticket.c
   ```
2. **These are the only files that you should modify!**
3. **You may add helper functions if you like, but they need to be located in the same file as the function that uses them as we are testing the files individually.**
4. Make sure that you remove the comment for the correct function in SELVERS.h

Suggested approach: Develop and test each function by itself with the solution versions for all the other functions. When you get all four replacement functions fully working independently, then test them all together.

Coding Requirements

1. You will write your code only in C and it must run in a Linux environment on the pi-cluster.
2. In the code you write you cannot use calloc(). The only use of calloc() in the program is the ones in the starter file main.c
3. **Do not use recursive approaches in your code.**
4. **You cannot use any downloaded software** (other than what is in the github repository **for this PA**). **If it is not already installed on the pi-cluster by ITS, you cannot use it.**
5. Make sure you pass all the tests before submitting to gradescope.

Error Messages

The only required error messages for insertticket() are for malloc failure cases.
Use the following:

```
fprintf(stderr, "%s: unable to allocate vehicle for summons %s\n", argv0, summ);
fprintf(stderr, "%s: unable to allocate state for summons %s\n", argv0, summ)
fprintf(stderr, "%s: unable to allocate plate for summons %s\n", argv0, summ);
fprintf(stderr, "%s: unable to allocate ticket for summons %s\n", argv0, summ);
```

There are no error messages required for delticket().

Turning in Files For Your Grade

Before submitting your program to gradescope, make sure at a minimum that your program passes the supplied tests in the test harness.

Submit to gradescope under the **assignment titled PA7**
ONLY submit the following files (total of two (2) files):

```
insticket.c
delticket.c
```

You can upload multiple files to Gradescope by holding CTRL (⌘ on a Mac) while you are clicking the files. You can also hold SHIFT to select all files between a start point and an endpoint. Alternatively, you can place all files in a folder and upload the folder to the assignment. Gradescope will upload all files in the folder. You can also zip all the files and upload the .zip to the assignment. Ensure that the files you submit are not in a nested folder.

Appendix 1 - Optional — What is this testP?

testP is a really simplistic performance metric on the database. Now the code is not being compiled with the optimizers on, but this still gives you an indication of how fast your code runs. The supplied solution code has a lot of additional tests to try to help you debug your code, so your routines should be as least as fast. The timings are written to stderr so they report as errors by the test harness even though they are not errors!
Here is a sample run:

```
cs30fa22@pi-cluster-153:~/PA6 % make testP

****** starting testP *******
./runtest P

----- Starting test number P -----
Running in/cmdP < in/testP > out/outP 2> out/errP
cmdP is: time ./parking -s -t409 -f in/Fines.csv -d in/Large.csv
./parking returned EXIT_SUCCESS
    Comparing exp/outP to out/outP
**** Standard out   on test number P passed ****
    Comparing exp/errP to out/errP

**** Standard error on test number P failed ****

*** exp/errP 2022-10-23 17:40:41.397440848 -0700
--- out/errP 2022-10-29 01:37:18.990930463 -0700
***************
*** 0 ****
--- 1,4 ----
+
+ real 0m37.104s
+ user 0m34.310s
+ sys  0m2.711s
----- Ending   test number P -----

******* All Done *******
```

Ignore the line Standard error on test number P failed that is normal for this test (it is not failing). The important numbers are the user and sys lines (smaller is better), real time is not a good performance guide.

Appendix 2 – Testing everything: make alltest output

Here is the output from run make alltest

```
cs30fa22@pi-cluster-153:~/PA6 % make alltest

****** starting testA *******
./runtest 1 2 DUP

----- Starting test number 1 -----
Running in/cmd1 < in/test1 > out/out1 2> out/err1
cmd1 is: ./parking -s -t3 -f in/Fines.csv -d in/Tiny.csv
./parking returned EXIT_SUCCESS
    Comparing exp/out1 to out/out1
**** Standard out   on test number 1 passed ****
    Comparing exp/err1 to out/err1
**** Standard error on test number 1 passed ****
----- Ending   test number 1 -----

----- Starting test number 2 -----
Running in/cmd2 < in/test2 > out/out2 2> out/err2
cmd2 is: ./parking -s -t3 -f in/Fines.csv -d in/Tiny.csv
./parking returned EXIT_SUCCESS
    Comparing exp/out2 to out/out2
**** Standard out   on test number 2 passed ****
    Comparing exp/err2 to out/err2
**** Standard error on test number 2 passed ****
----- Ending   test number 2 -----

----- Starting test number DUP -----
Running in/cmdDUP < in/testDUP > out/outDUP 2> out/errDUP
cmdDUP is: ./parking -s -t3 -f in/Fines.csv -d in/Dups.csv
./parking returned EXIT_SUCCESS
    Comparing exp/outDUP to out/outDUP
**** Standard out   on test number DUP passed ****
    Comparing exp/errDUP to out/errDUP
**** Standard error on test number DUP passed ****
----- Ending   test number DUP -----

******* All Done *******

****** starting testV1 *******
./runtest V1

----- Starting test number V1 -----
Running in/cmdV1 < in/testV1 > out/outV1 2> out/errV1
cmdV1 is: valgrind -q --leak-check=full --leak-resolution=med -s ./parking -s -t3 -f in/Fines.csv
-d in/Tiny.csv 2>&1 1>/dev/null | cut -f2- -d' ' 1>&2 2>/dev/null
```

```
./parking returned EXIT_SUCCESS
    Comparing exp/outV1 to out/outV1
**** Standard out   on test number V1 passed ****
    Comparing exp/errV1 to out/errV1
**** Standard error on test number V1 passed ****
----- Ending   test number V1 -----

******* All Done *******

****** starting testB *******
./runtest 3 4

----- Starting test number 3 -----
Running in/cmd3 < in/test3 > out/out3 2> out/err3
cmd3 is: ./parking -s -t3 -f in/Fines.csv -d in/Small.csv
./parking returned EXIT_SUCCESS
    Comparing exp/out3 to out/out3
**** Standard out   on test number 3 passed ****
    Comparing exp/err3 to out/err3
**** Standard error on test number 3 passed ****
----- Ending   test number 3 -----

----- Starting test number 4 -----
Running in/cmd4 < in/test4 > out/out4 2> out/err4
cmd4 is: ./parking -s -t3 -f in/Fines.csv -d in/Small.csv
./parking returned EXIT_SUCCESS
    Comparing exp/out4 to out/out4
**** Standard out   on test number 4 passed ****
    Comparing exp/err4 to out/err4
**** Standard error on test number 4 passed ****
----- Ending   test number 4 -----

******* All Done *******

****** starting testC *******
./runtest 5

----- Starting test number 5 -----
Running in/cmd5 < in/test5 > out/out5 2> out/err5
cmd5 is: ./parking -s -t409 -f in/Fines.csv -d in/Medium.csv
./parking returned EXIT_SUCCESS
    Comparing exp/out5 to out/out5
**** Standard out   on test number 5 passed ****
    Comparing exp/err5 to out/err5
**** Standard error on test number 5 passed ****
----- Ending   test number 5 -----

******* All Done *******
```

```
****** starting testD *******
./runtest 6

----- Starting test number 6 -----
Running in/cmd6 < in/test6 > out/out6 2> out/err6
cmd6 is: ./parking -s -f in/Fines.csv -d in/Large.csv
./parking returned EXIT_SUCCESS
    Comparing exp/out6 to out/out6
**** Standard out   on test number 6 passed ****
    Comparing exp/err6 to out/err6
**** Standard error on test number 6 passed ****
----- Ending   test number 6 -----

******* All Done *******

****** starting testV2 *******
./runtest V2

----- Starting test number V2 -----
Running in/cmdV2 < in/testV2 > out/outV2 2> out/errV2
cmdV2 is: valgrind -q --leak-check=full --leak-resolution=med -s ./parking -s -t409 -f
in/Fines.csv -d in/Medium.csv 2>&1 1>/dev/null | cut -f2- -d' ' 1>&2 2>/dev/null
./parking returned EXIT_SUCCESS
    Comparing exp/outV2 to out/outV2
**** Standard out   on test number V2 passed ****
    Comparing exp/errV2 to out/errV2
**** Standard error on test number V2 passed ****
----- Ending   test number V2 -----

******* All Done *******
```