CSE 1320: Intermediate Programming
University of Texas at Arlington
Fall 2021
Alex Dillhoff

Assignment 6

This assignment focuses on stacks, queues, and hash maps.

Your code must compile without any warnings or errors and run without segmentation faults to receive credit. Additionally, any allocated memory must be freed. Points will be taken for inconsistent formatting.

1. An automotive repair shop has contracted you to develop an application which will allow them to properly track their pending work base on a First In First Out (FIFO) system. Create a program that allows the user to enter the information of a vehicle based on the data structure given below. The vehicle will be added to the queue.

   The main menu should have an option to either add a new vehicle to the queue or view the next vehicle to be repaired. If they choose to view the next vehicle, present a sub-menu that asks if they want to start repairs or go back. If they choose to start the repair, remove the vehicle from the queue and go back to the main menu.

   Data Format

   ```
   typedef struct {
       int year;
       char *make;
       char *model;
       char *color;
       char *license;
   } vehicle_t;
   ```

   Other Requirements

   - Allocated memory must be freed before the program terminates.
   - Any function declarations, `struct` declarations, and library includes should be in a corresponding header file.
   - Save your files as `problem1.h` and `problem1.c`.

   Example Run

   ```
   1. Add Vehicle
   2. View Next Vehicle
   3. View Current Queue
   4. Quit
   > 2
   Next up: 1999 Nissan Skyline GT-R R34 (Silver) LIC#T4U842
   ```

```
1. Start Repair
2. Go back
> 2
1. Add Vehicle
2. View Next Vehicle
3. View Current Queue
4. Quit
> 3
1. 1999 Nissan Skyline GT-R R34 (Silver) LIC#T4U842
2. 2002 Mitsubishi Lancer Evolution VII (Lime Green) LIC#Z3Y921
3. 2000 Honda S2000 (Hot Pink) LIC#W3E566
4. 2003 Mitsubishi Eclipse Spyder GTS (Purple) LIC#H8TR
5. 1970 Dodge Challenger R/T (Orange) LIC#D2X437
```

2. Create a program that reads in a stack trace via text file and prints the output to `stdout`. When a function call is read in the file, it should be pushed to a stack. When a `return` is read, it should pop the last item off of the stack (see example output).

   The output should be printed such that the scope of each function call is apparent.

   Other Requirements

   - The text file must be input as a command line argument when running the program.
   - Allocated memory must be freed before the program terminates.
   - Any function declarations, `struct` declarations, and library includes should be in a corresponding header file.
   - Save your files as `problem2.h` and `problem2.c`.

   Example Run

   Based on the sample file given in Canvas.

```
main()
    preprocess()
    train()
        training_loop()
            run_batch()
        validate()
    test()
    save_results()
```

3. Implement a hash map that uses separate chaining via dynamic arrays and incremental rehashing. The hash map should store vehicle values using the same data format as in problem 1. The program should allow users to add vehicles from `stdin` as well as the ability to import from a file.

   The keys for this map will be the license plates of each vehicle.

   Other Requirements

- When importing a file to add, gather a collection of unique keys to add before using any hash map operations. Once the unique keys are gathered, then rehash and resize with at least enough room to store the new keys without rehashing again.
- When adding a vehicle, check to make sure the vehicle is not already in the hash map. If it is, warn the user and return to the main menu.
- Allocated memory must be freed before the program terminates.
- Any function declarations, `struct` declarations, and library includes should be in a corresponding header file.
- Save your files as `problem3.h` and `problem3.c`.

Example Output

```
1. Add Vehicle
2. Import File
3. Vehicle Lookup
4. Print Map
> 2
Enter filename: cars.csv
14 Entries to be entered.
Rehashing...
Import complete.
```

Create a `zip` file using the name template `LASTNAME_ID_A6.zip` which includes the all required code files. Submit the `zip` file through Canvas.