CIS 2520 Data Structures

Assignment#2 Guidelines

Fall 2009

<u>**Assignment 2 is due Tuesday October 20 by 11:59:59 pm.**</u>

- Submit Assignment 2 through your account in MOODLE. **All source files, a readme file, test files and makefile should be included in one folder and submitted as a zip/tar file.**

- Submission file name should include only your lastname, ID for instance: **Smith_0619866.zip or Smith_0619866.zip**

- You are allowed to use standard library functions. The tar/zip file should contain:

  readme.txt
  calculator.c
  makefile
  the file(s) for question2

- For the makefile, all programs should be compiled with -Wall and -ansi. The first line of the makefile should be an all rule, as the programs will be compiled using make with no arguments.

- <u>**Late Submission Penalty: For every 30 minutes 5% marks will be deducted.**</u>

- <u>**Submission of all source file as a one document file will be marked zero**</u>

- If the makefile does not successfully compile your programs, marks will be deducted. Any warnings will result in a mark deduction appropriate to the severity of the warnings. There will be some marks allocated for style and documentation, but the majority will come from the correctness of the programs.

- List all the limitations of your program and everything you want to share with TA who marks the assignment in readme.txt.

- Remember to include your name and student ID in all files.

Question 1:

(1) Your program should handle input such as "12+" and "1 2 +". In "1 2 +", there is exactly one space in between, not no space before or after the expression.

(2) You are NOT expected to handle unary operators such as negation. You only need to handle four operations: + - * /.
(3) It's ok if the results of intermediate steps are integers. For example, "1 3 5 / +" should give 1.6 as final result. However, you will get full marks if you have 1 or 2 as the final result.
(4) Your program should ask user (TA) to input the postfix notation infinite times until character 'q' is entered. Program terminates when 'q' is entered.
(5) Organize your output reasonable, meaningful and easy to read.
(6) You can use any implementation of stack. Linked list, array, etc. However, you need to think about the size of stack. Make the capacity of the stack reasonable.
(7) Error messages are expected to be general but meaningful. You do NOT need to distinguish "9 +" (too few operands) and "9 9 9 +" (too many operands). However, if you pop from an empty stack, you should have a specific message corresponding to this case. Be creative and think about the errors in mathematics.
(8) Your program will be tested in linux machine by the following command:

```
./calculator < test1.txt
```

This means your executable should be named calculator, and a sample test.txt file is provided. The test file used for marking is going to be more complicated so that all the cases will be tested.

Question 2:

(1) New car does NOT necessary to have a mileage of 0.
(2) Return the car to available-for-rent list or to repair list, means you need to calculate the cost.
(3) Transfer a car from repair list to available-for-rent list is not going to calculate cost.
(4) No order is required for repair list.
(5) Valid license numbers are of the form "ABCDE". No integers in license numbers. All 5 letters are in capital case. You do NOT need to handle errors such as more than one car with the same license.
(6) Valid dates are: YYYY [2000, 2020], MM [01, 12], DD [01, 31]. To simplify the implementation, we assume 20100231 is a valid return date.
(7) You can assume the correct procedures are add a new car(1), rent the car(5),and then return the car(2 or 3). Therefore, if you try to return a car which is not in the rented list, you need to output an error message. Error messages should be generated for such illegal operations.
(8) Organize your output reasonable, meaningful and easy to read.
(9) It is better to use just one file to implement this question. If you used more than one, that's ok. However, your program will be tested as follows:

```
./car < test2.txt
```

Provide me all your test files in the Submission. Please note again, there is a mark allocated to program style, good documentation (readme).