National University of Singapore
School of Computing

CS2281 — **Practical Test 1** — Semester 3 04/05

Instructions

PLEASE READ THE INSTRUCTIONS CAREFULLY.

- You have THREE (3) hours to complete this test.
- This is an OPEN book exam, you may refer to your books, notes or any resources online.
- You must answer exam questions independently and may NOT try to obtain outside help of any kind.
- TURN OFF your handphone.
- Answer ALL questions.
- This paper contains FOUR (4) printed pages, including this cover page.

Procedure

- Login to the PC using your NUSNET account as usual.
- You should have received a user id and a password for your special exam account from your invigilator. You must login to sunfire (also known as sf3) using this account.
- **Your code must remain in your special exam account all the time. Transferring of your code to other systems (including laptop, desktop, mail server, web server, file server) is STRICTLY prohibited.**
- You are encouraged to save your work often.
- At the end of this exam, save your files, leave them in your account, and log off. We will collect your files from your exam account after the exam.

Question 1 (50 points)

Write a C program called `q1.c` that accepts integers as command line arguments and print their average to `stdout` as a floating point number (up to 3 decimal places). For example,

```
$ q1 3 4
3.500
$ q1 0 1 -1 2
0.500
```

The user might input command line arguments that are not valid integers. Your program should detect these invalid arguments and ignore them when computing the average.

For the purpose of this question, we define integer to be a string which consists of only characters between '0' and '9', except the first character which may either be the '-' character or the '+' character. If the first character is '-' or '+', then it must be followed by at least one other characters between '0' and '9'. For instance, "-7", "007", "789", "+789", "-0" are all valid integers, but "-", "4.0", "abc" are not valid integers.

Since `atoi` does not check for valid inputs, you are required to write a function that checks the validity of an input argument before passing it to `atoi` for conversion from `char *` to `int`. The function checks if an input argument is a valid integer and has the following prototype.

```
int is_integer(char *s)
```

It should takes in a string `s` and returns 1 if `s` is a valid integer as defined above. Otherwise, `is_integer` must return 0. (Note: You cannot change the function prototype for `is_integer()`)

Inputs that are not valid integers must be detected by your function `is_integer` and ignored when calculating the average. Print a warning message to `stderr` to warn the user about the errors in the input. Your warning message must follow the format shown in the example below. In particular, it must contain the invalid command line arguments.

```
$ q1 9 10 a 3.1415 11
warning: "a" "3.1415" are ignored because they are not integer.
10.000
```

If no valid integer argument is passed in, the average is taken to be 0.

```
$ q1 one two three
warning: "one" "two" "three" are ignored because they are not integer.
0
```

Additional Tips

- To check if a character `c` is a numeric character (between '0' and '9', you can use `(c >= '0' && c <= '9')`.
- Do not assume maximum number of command line arguments you can pass into your `main` function.

Question 2 (50 points)

In this question, you are asked to implement a file-based simple database that stores records about students. The database will be implemented as a *text* file where each line is a record. You will write a C program named `q2.c` that manipulates the student records stored in the text file. A student record consists of a *name*, *userid*, *age* and *CAP*. *name* can be represented as a string which is at most 32 characters in length. *userid* is a string which is at most 8 characters in length. You can assume that a userid uniquely identifies a student. *age* is an `int` and *CAP* can be represented using a `float`. You may also assume that there will be no more than 256 records in the database.

Your program should manipulate the database using three operations, *add*, *find* and *delete*. Usage specification for your program is given below.

- q2 add *db name userid age cap*
  Append a record specified by *name*, *userid*, *age* and *cap* to the end of text file named *db*. If the database *db* does not exist, create a new file. Example usage:

  ```
  $ q2 add student.db "Tan Ah Kow" tanahkow 19 4.5
  ```

  You may assume the given userid does not already exist in the database and the inputs are always valid (*name* will never be longer than 32 characters, *age* is a valid integer, and *cap* is a valid floating point number). Note that by putting quotes around "Tan Ah Kow", Tan Ah Kow will be treated as one single argument rather than three.

- q2 find *db userid*
  Find the student record with *userid* from database *db*. If the record cannot be found, print nothing. Otherwise, print out the name, userid, age and cap of the student as formatted below.

  ```
  $ q2 find student.db tanahkow
  Tan Ah Kow tanahkow 19 4.5
  ```

- q2 delete *db userid*
  Find the student record with *userid* from database *db*. If the record cannot be found, print an appropriate error. Otherwise, delete the first record for student *userid* from *db*.

  ```
  $ q2 delete student.db tanahkow
  $ q2 delete student.db tanahkow
  No such student tanahkow in student.db
  ```

- If the first command line argument is neither add, delete nor find, or if total number of arguments does not match, print an appropriate error message.

Additional Tips

- To append to a file, open the file with mode "a".
- You may format the text file in any way you want. Pay special care to how names are stored – Since a name can contain spaces, you cannot use `fscanf(f, "%s", name)` to read in a name. One trick is to store names as fixed length string[1], and store the length of the names in the text file as well. Another trick is to replace spaces in names with special characters ("Tan Ah Kow" becomes "Tan_Ah_Kow") so that you can use scanf to read in the whole name. Yet another trick is to use `strspn()` to trim the leading white spaces. You can use which ever method you want, as long as the output is formatted properly.
- To delete a line, you may use the following simple but inefficient method: open the file for reading, read all the lines and store it somewhere, close the file, open the file again for writing, write everything back except the deleted line and close the file.

[1]For a fixed length of 10, use `printf` with `%10s` to write, and `scanf` with `%10c` to read. Read man page for details.