COMP281 Assignment 3 Resit

- In the following, you will find 1 problem that constitute the Assignment 3 Resit. It will be also available on the *online judging (OJ)* system available at http://intranet.csc.liv.ac.uk/JudgeOnline/

- You need to write a valid C program that solves each of these problems – it must read the input, as specified in the problem description then print the solution to the given problem for that input.
  - Note that code is "correct" only if it **correctly implements a solution to the problem stated** in the assignment, not "if online judge accepts it".
  - That is, even if OJ accepts your code, it could be wrong. Read the problems carefully.

- Input is read from the standard input, in the same way that you read input from the keyboard as shown in lectures (e.g., using scanf). Output is also printed to the standard output, as you have seen (e.g., using printf).

- When you are satisfied that your programs work correctly, you must submit them through the departmental submission system, as described below.
  - Even if the program is not correct, still submit whatever you have! You can still earn points if certain parts of the code are done right.

- You must also include a brief report describing your solutions to the problems. This should be maximum one side of A4 paper (although there is no penalty for a longer report) and should give a description of how each of your solutions works. This should include describing the algorithm used to reach the solution, describing your use of any C language features (that were not discussed in lectures) and identifying any resources that you have used to help you solve the problems.

- This assignment is worth 40% of the total mark for COMP281.
  - All problems are weighted equally.
  - For each problem, you can earn a total of 20 points
    - 10 points for "Functionality and Correctness" awarded for programs that **correctly** solve the problem for all test cases.
    - 8 points for "Programming style, use of comments, indentation and identifiers" awarded depending on the style, comments and efficiency of the solution
    - 2 points for the quality and depth of the accompanying report
  - The final grade results from normalizing the earned points to a scale of 100.
  - See separate "COMP 281 Detailed Marking Guidelines" for more details.

Submission Instructions

- Name each of your c files according to the problem number; i.e., problem 10xx should be in a file 10xx.c (where 'xx' is replaced by the actual number).
- Place all of your C files, one per problem, and your report (in .pdf format) into a single zip file.
    - Please use the standard (pkzip) zip file format:
      https://en.wikipedia.org/wiki/Zip_%28file_format%29
      which is supported by winzip, winrar, etc. on windows/mac os X, and 'zip' on linux
    - test your zip file before submitting.
- Before you submit your solutions, please first test them using the online judge. **You are required to include the "Result" and the "RunID" in your C code as comments.** The OJ provides a RunID. **RunIDs are not problem IDs**.
    - Example:
        - the problem is 10xx
        - The solution has been accepted by the OJ, and the runID is 2033.
        - You add to your code: /* 2033 Accepted */ to 10xx.c
    - Result is one of the following: Accepted, Wrong Answer, Presentation Error, Time Limit Exceeded, Memory Limit Exceeded, Output Limit Exceeded, Runtime Error, Compile Error
- Submit this zip file using the departmental submission system at
  http://www.csc.liv.ac.uk/cgi-bin/submit.pl
  Only the file submitted through this link will be marked.
- The **deadline** for this assignment is **11th August 2017, 5pm.**
- Penalties for late submission apply in accordance with departmental policy as set out in the student handbook, which can be found at:
  http://intranet.csc.liv.ac.uk/student/ug-handbook.pdf

1. Problem 1077

Title: Birthday Lookup

Description

In this assignment, you will need to create a 'database' of birthdays that will be queried repeatedly.

First, a (non-specified) number of birthdays + names is specified. You will need to store this data in a binary search tree (BST). The nodes of the BST will contain the (birthday, name) pairs.

Once the birthdays are stored, a number of queries follows. These take the shape of numeric (D-M-YY) dates. For each queried date you need to return the person that has a birthday on that date, or, if the queried date is nobodies birthday, the name of the person whose birthday is the first after the query date.

- Each line corresponds to a **valid** date, and a name. (You do not have to worry about date validation; all dates in the input are valid dates.)
- Each date consisting of one string ("January", "February", ..., or "December"), one integer between 1 and 31, and one two digit integer representing the year (from 90 to 99, and then from 00 to 16).
- Each name consists of a first name and a last name. (i.e., you do not need to worry about compound names comprised of more than 2 'words')
- Please use structures to store the dates and names.
- Use malloc to dynamically allocate just enough space for all the data and data structures.
- The access time of binary trees depends on how "balanced" they are. So a perfect solution would deal with this issue.

Input

- the string "BIRTHDAYS_START"
- followed by the (date, birthday)-pairs
- followed by the string "QUERIES_START"
- followed by a number of user query in format *day month year* (e.g. "1 1 00" or "31 3 68").

Output

- the names of persons who have their birthday on the queried dates, or those of the first subsequent birthdays.
- In the latter case, this should be indicated by pre-pending "first subsequent birthday: " before the name.
- In the unlikely case that there is no next birthday, you should print "no subsequent birthday"

Sample Input

```
BIRTHDAYS_START
January 1 01 Molly Mcauliffe
January 1 00 Dennise Nigh
February 28 99 Erma Merrick
July 17 12 Linn Alvin
September 10 12 Delphia Bynum
July 1 00 Vania Jones
June 30 90 Brittney Gemmill
August 25 06 Aubrey Sherard
May 27 08 Marica Rising
October 1 03 Eugena Steele
QUERIES_START
1 1 00
3 7 12
30 6 90
6 7 16
1 1 90
```

Sample Output

```
Dennise Nigh
first subsequent birthday: Linn Alvin
Brittney Gemmill
no subsequent birthday
first subsequent birthday: Brittney Gemmill
```