CSCI3180 – Principles of Programming Languages – Spring 2020

Assignment 1 — TA Ranking System in COBOL and C

Deadline: Feb 09, 2019 (Sunday) 23:59

1 Introduction

Teachers always want to find the best and dedicated TAs for their courses. Different courses require different skills sets. Different TAs have different skills and also preferences for the courses that they are interested in. In this assignment, you have to implement part of a TA matching system. Your program should help each course to rank the top 3 TAs. You are required to implement it once using COBOL and once using C.

2 Assignment Details

Jimmy is designing a simple TA Matching System (the "System" hereafter) to find the most suitable TAs for each course. He is asking you to implement the TA ranking module of this system. The system reads 1) the courses' requirements as well as 2) the candidate TAs' skills and preferences as input, and reports the top 3 TAs for each course, where TAs are ranked by *matching scores* (to be explained later).

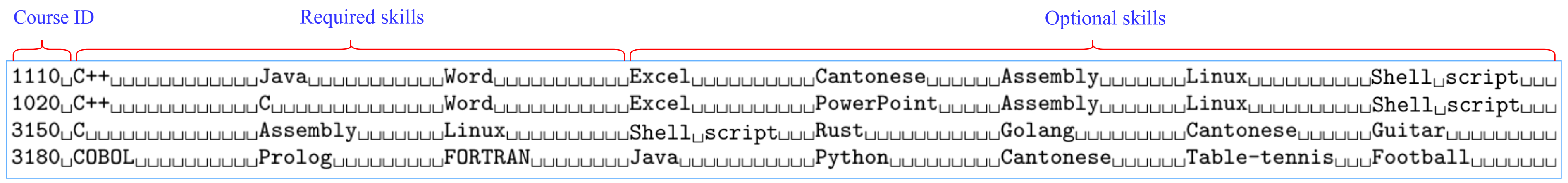
2.1 TA ranking system

In this System, course instructors will enter their requirements for TAs: 3 required skills, and 5 optional skills. Candidate TAs will enter their profiles: 8 skills, and their 1st, 2nd and 3rd preferred courses.

The System keeps the course instructors' requirements in a file: **instructors.txt**. Each line of the file represents the requirements of a course, containing the course ID, required skills, and optional skills. Also, the System keeps another file containing all the candidate TAs' information: **candidates.txt**. Each line of the file records a candidate TA's information, including the TA ID, skills, and preferred courses. Figures 1 and 2 give an example of the **instructors.txt** and **candidates.txt** files (spaces are explicitly displayed as ␣) respectively. Notice that skill names are padded by spaces so that each skill name takes exactly 15 characters. Some skill names may contain multiple words and have spaces between words, e.g. `Shell␣script␣␣␣`.

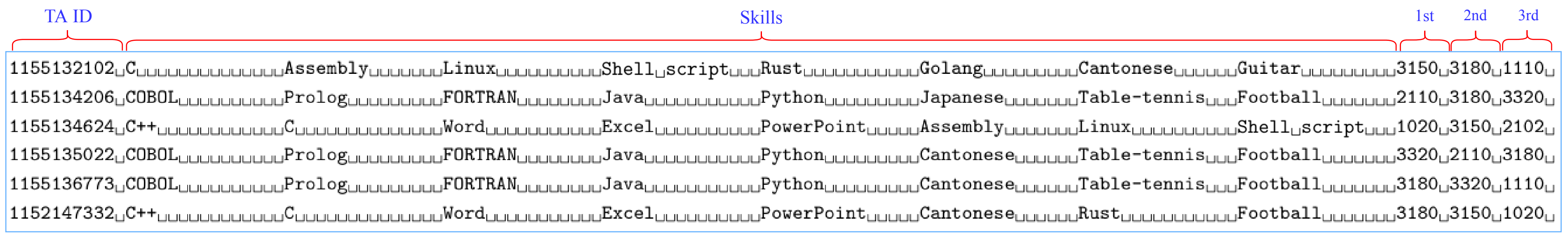
Jimmy has designed the following policy regarding the matching score that measures how suitable a candidate is to be the TA of a course. The matching score of each pair of (*course, TA*) is computed as:

$$score(course, TA) = \begin{cases} 1 + skill\_score + preference\_score & \text{if all the required skills are satisfied} \\ 0 & \text{otherwise} \end{cases}$$


Course ID
Required skills
Optional skills

```
1110␣C++␣␣␣␣␣␣␣␣␣␣␣␣Java␣␣␣␣␣␣␣␣␣␣␣Word␣␣␣␣␣␣␣␣␣␣␣Excel␣␣␣␣␣␣␣␣␣␣Cantonese␣␣␣␣␣␣Assembly␣␣␣␣␣␣␣Linux␣␣␣␣␣␣␣␣␣␣Shell␣script␣␣␣
1020␣C++␣␣␣␣␣␣␣␣␣␣␣␣C␣␣␣␣␣␣␣␣␣␣␣␣␣␣Word␣␣␣␣␣␣␣␣␣␣␣Excel␣␣␣␣␣␣␣␣␣␣PowerPoint␣␣␣␣␣Assembly␣␣␣␣␣␣␣Linux␣␣␣␣␣␣␣␣␣␣Shell␣script␣␣␣
3150␣C␣␣␣␣␣␣␣␣␣␣␣␣␣␣Assembly␣␣␣␣␣␣␣Linux␣␣␣␣␣␣␣␣␣␣Shell␣script␣␣␣Rust␣␣␣␣␣␣␣␣␣␣␣Golang␣␣␣␣␣␣␣␣␣Cantonese␣␣␣␣␣␣Guitar␣␣␣␣␣␣␣␣␣
3180␣COBOL␣␣␣␣␣␣␣␣␣␣Prolog␣␣␣␣␣␣␣␣␣FORTRAN␣␣␣␣␣␣␣␣Java␣␣␣␣␣␣␣␣␣␣␣Python␣␣␣␣␣␣␣␣␣Cantonese␣␣␣␣␣␣Table-tennis␣␣␣Football␣␣␣␣␣␣␣
```

Figure 1: Sample of instructors.txt


TA ID
Skills
1st
2nd
3rd

```
1155132102␣C␣␣␣␣␣␣␣␣␣␣␣␣␣␣Assembly␣␣␣␣␣␣␣Linux␣␣␣␣␣␣␣␣␣␣Shell␣script␣␣␣Rust␣␣␣␣␣␣␣␣␣␣␣Golang␣␣␣␣␣␣␣␣␣Cantonese␣␣␣␣␣␣Guitar␣␣␣␣␣␣␣␣␣3150␣3180␣1110␣
1155134206␣COBOL␣␣␣␣␣␣␣␣␣␣Prolog␣␣␣␣␣␣␣␣␣FORTRAN␣␣␣␣␣␣␣␣Java␣␣␣␣␣␣␣␣␣␣␣Python␣␣␣␣␣␣␣␣␣Japanese␣␣␣␣␣␣␣Table-tennis␣␣␣Football␣␣␣␣␣␣␣2110␣3180␣3320␣
1155134624␣C++␣␣␣␣␣␣␣␣␣␣␣␣C␣␣␣␣␣␣␣␣␣␣␣␣␣␣Word␣␣␣␣␣␣␣␣␣␣␣Excel␣␣␣␣␣␣␣␣␣␣PowerPoint␣␣␣␣␣Assembly␣␣␣␣␣␣␣Linux␣␣␣␣␣␣␣␣␣␣Shell␣script␣␣␣1020␣3150␣2102␣
1155135022␣COBOL␣␣␣␣␣␣␣␣␣␣Prolog␣␣␣␣␣␣␣␣␣FORTRAN␣␣␣␣␣␣␣␣Java␣␣␣␣␣␣␣␣␣␣␣Python␣␣␣␣␣␣␣␣␣Cantonese␣␣␣␣␣␣Table-tennis␣␣␣Football␣␣␣␣␣␣␣3320␣2110␣3180␣
1155136773␣COBOL␣␣␣␣␣␣␣␣␣␣Prolog␣␣␣␣␣␣␣␣␣FORTRAN␣␣␣␣␣␣␣␣Java␣␣␣␣␣␣␣␣␣␣␣Python␣␣␣␣␣␣␣␣␣Cantonese␣␣␣␣␣␣Table-tennis␣␣␣Football␣␣␣␣␣␣␣3180␣3320␣1110␣
1152147332␣C++␣␣␣␣␣␣␣␣␣␣␣␣C␣␣␣␣␣␣␣␣␣␣␣␣␣␣Word␣␣␣␣␣␣␣␣␣␣␣Excel␣␣␣␣␣␣␣␣␣␣PowerPoint␣␣␣␣␣Cantonese␣␣␣␣␣␣Rust␣␣␣␣␣␣␣␣␣␣␣Football␣␣␣␣␣␣␣3180␣3150␣1020␣
```

Figure 2: Sample of candidates.txt

where

- *skill_score*: number of optional skills satisfied by the TA
- *preference_score*: TA's preference to the course, according to the following table

	1st preference	2nd preference	3rd preference
preference_score	1.5	1	0.5

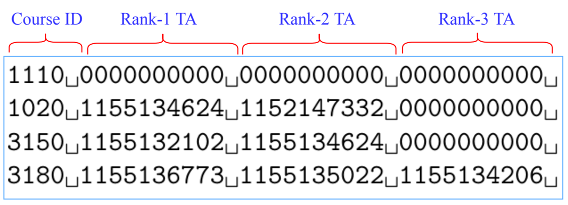
Taking the **instructors.txt** and **candidates.txt** in Figures 1 and 2 respectively as an example:

$$score(3180, 1155136773) = 1 + 5 + 1.5 = 7.5$$

$$score(3180, 1152147332) = 0$$

The **output.txt** file should not output the matching scores of every TA for each course directly, but reports only the top 3 TAs for each course. Figure 3 is an example of the **output.txt** file. Note that, if less than k candidates satisfy the required skills specified by the course, the Rank-k TA for that course is filled with `0000000000␣`:

Besides, pay attention to the cases that the **instructors.txt** file or the **candidates.txt** file is empty. If the **instructors.txt** file is empty, the **output.txt** file should be empty too; if the **candidates.txt** file is empty, the Rank-k TA for all the courses should be filled with `0000000000␣`.


Course ID
Rank-1 TA
Rank-2 TA
Rank-3 TA

```
1110␣0000000000␣0000000000␣0000000000␣
1020␣1155134624␣1152147332␣0000000000␣
3150␣1155132102␣1155134624␣0000000000␣
3180␣1155136773␣1155135022␣1155134206␣
```

Figure 3: Sample of output.txt

Jimmy needs your help. You are required to implement a program to generate the ranking report of each course.

2.2 General Specification

You are required to write two programs, one in COBOL and the other one in C, for the student ranking module of the System. You should name your COBOL source as "ta_ranking.cob" and your C source as "ta_ranking.c".

1. **Input and Output Specification**
   Your programs should read two input files: **instructors.txt** and **candidates.txt**, which contain instructors' requirements and candidates' skills and preferences. The detailed specification of input format is given in Section 2.3. For each course, your program needs to calculate the matching score for each candidate. These calculations should be strictly based on the policy described above. Afterwards, TAs are ranked by their matching scores with each course. In case of a tie (two TAs having the same matching score), the TA with **lower** TA ID will have a better rank than the other. Your program should output one file to report

the top 3 TAs for each course and display them in ranking order (i.e. the first one is the best TA, the second one is the second best and so on). The output file format should follow the description in Section 2.4. You can "hardcode" the input file names in your program. And the naming of the output file should be: **output.txt**.

2. **Restrictions on using COBOL and C**
   For COBOL, in order to force you to program as in the old days, ONLY 2 keywords are allowed in selection and loop statements: "IF" and "GOTO". You are not allowed to use modern control constructs, such as if-then-else or while loop. Using any other keywords will receive marks deduction. But for C, you can use whatever you want.

3. **Error Handling**
   The programs should also handle possible errors gracefully by printing meaningful error messages to the standard output. For example, your program should be able to check whether the input file exists or not. If not, display a warning message "non-existing file!". However, you **CAN** assume that the input files are free of format on content errors.

4. **Good Programming Style**
   A good programming style not only improves your grade but also helps you a lot in debugging. Poor programming style will receive marks deduction. Construct your program with good readability and modularity. Provide sufficient documentation by commenting your codes properly but never redundantly. Divide up your programs into subroutines instead of clogging the main program. The main section of your program should only handle the basic file manipulation such as file opening and closing, and subprogram calling. The main purpose of programming is not just to make the program right but also make it good.

5. **Other Notes**
   You are **NOT** allowed to implement your program in another language (e.g. Java/Python) and then initiate system calls or external library calls in COBOL and C. Your source codes will be compiled and PERUSED, and the object code tested!

   Do not implement your programs in multiple source files. Although COBOL and C do allow you to build a project with subroutines scattered among multiple source files, you should only submit one source file for each language.

   NO PLAGIARISM!!!! You are free to design your own algorithm and code your own implementation, but you should not "borrow" codes from your classmates. If you use an algorithm or code snippet that is publicly available or use codes from your classmates or friends, be sure to DECLARE it in the comments of your program. Failure to comply will be considered as plagiarism.

   A crash introduction to COBOL will be given in the upcoming tutorials. Please DO attend the tutorials to get a brief idea on COBOL, and then learn the language by yourselves. We assume that you are proficient in C. For a more in-depth study, we encourage students to search relevant resources on the Internet (just Google it!).

2.3 Input File Format Specification

There are two input files: **instructors.txt**, and **candidates.txt**. All input files are in plain ASCII text. Each line is ended with the characters "\r\n" on windows machine, including the last line. You can write your programs on whatever OS you like, but please verify that they can be compiled and run correctly on Windows machines in SHB924/909 because we will grade your assignment there. You should strictly follow the format as stated in the following.

- Each line of **instructors.txt** contains nine fields of fixed length for a course. The lines are sorted by course IDs in **ascending order**.

  1. *Course ID*: a 4-digit number followed by a space.

2. *Required skills*: 3 required skills, each of which is a string of 15 characters (spaces are padded at the end in case the skill name is less than 15 characters).
3. *Optional skills*: 5 optional skills, each of which is a string of 15 characters (spaces are padded at the end in case the skill name is less than 15 characters).

- Each line of **candidates.txt** contains twelve fields of fixed lengths of a candidate's skills and preferences. The lines are sorted by TA IDs in **ascending order**.

  1. *TA ID*: a 10-digit number followed by a space.
  2. *Skills*: 8 skills, each of which is a string of 15 characters (spaces are padded in case the skill name is less than 15 characters).
  3. *Preference*: 3 preferences, each of which is a 4-digit number followed by a space.

  You may make the following assumptions on the files:

  – All input files strictly follow the format specified in Section 2.3.
  – File **instructors.txt** is sorted by course ID in ascending order.
  – File **candidates.txt** is sorted by TA ID in ascending order.
  – The number of skills in file **candidates.txt** is fixed to 8, and there are no duplicate skills.

2.4 Output File Format Specification

There is only one output file: **output.txt**. You should strictly follow the format as stated in the following.

- Each line of **output.txt** contains 4 fields of fixed lengths of the top 3 TAs for each course. The lines should be sorted by course IDs in ascending order (the same order as in "**instructors.txt**").

  1. *Course ID*: a 4-digit number followed by a space.
  2. *Rank-1 TA*: The best TA, which is a 10-digit number followed by a space. If no candidates satisfy the required skills specified by the course, this field is filled by `0000000000␣`.
  3. *Rank-2 TA*: The 2nd best TA, which is a 10-digit number followed by a space. If less than 2 candidates satisfy the required skills specified by the course, this field is filled by `000000000␣`.
  4. *Rank-3 TA*: The 3rd best TA, which is a 10-digit number followed by a space. If less than 3 candidates satisfy the required skills specified by the course, this field is filled by `0000000000␣`.

  Pay attention to the special cases, as we have mentioned in the previous sections:

  – For Rank-k TA of a course, if less than k candidates satisfy the required skills specified by the course, the Rank-k TA for that course is filled with `0000000000␣`.
  – If the **instructors.txt** file is empty, the **output.txt** file should be empty too.
  – If the **candidates.txt** file is empty, the Rank-k TA for all the courses should be filled with `0000000000␣`.
  – If the input file is non-existent, display a warning message "non-existing file!".

2.5 Report

You should give a simple report to answer the following questions within one A4 page:

1. Compare the conveniences and difficulties in implementing the TA Ranking System in COBOL and C. You can divide the implementation into specific tasks such as "reading file in certain format", "simulating loops", "procedure/function call" and so on. Give code segments in your programs to support your explanation.
2. Compare COBOL with modern programming languages (e.g. Java/Python/...) from different aspects (e.g. variable declarations, paradigm, data type, parameter parsing, ...). You are free to pick your favorite modern programming language.
3. Do you think COBOL is suitable for writing applications like in this assignment, especially when some of the input needs to be passed several times? Explain in terms of, say, the aspect like programming difficulty, efficiency of you program, etc.
4. In your program design, how do you separate the tasks into submodules? Tell us briefly the functionality of each submodule and the main flow of your program in terms of these submodules.

3 Submission Guidelines

Please read the guidelines CAREFULLY. If you fail to meet the deadline because of submission problem on your side, marks will still be deducted.

The late submission policy is as follows:

- 1 day late: -20 marks

- 2 days late: -40 marks

- 3 days late: -100 marks

So please start your work early!

1. In the following, **SUPPOSE**

   your name is *Chan Tai Man*,
   your student ID is *1155234567*,
   your username is *tmchan*, and
   your email address is *tmchan@cse.cuhk.edu.hk*.

2. In your source files, insert the following header. REMEMBER to insert the header according to the comment rule of COBOL and C.

```
/*
 * CSCI3180 Principles of Programming Languages
 *
 * --- Declaration ---
 *
 * I declare that the assignment here submitted is original except for source
 * material explicitly acknowledged.  I also acknowledge that I am aware of
 * University policy and regulations on honesty in academic work, and of the
 * disciplinary guidelines and procedures applicable to breaches of such policy
 * and regulations, as contained in the website
 * http://www.cuhk.edu.hk/policy/academichonesty/
 *
 * Assignment 1
 * Name : Chan Tai Man
 * Student ID : 1155234567
 * Email Addr : tmchan@cse.cuhk.edu.hk
 */
```

The sample file header is available at

`http://course.cse.cuhk.edu.hk/~csci3180/resource/header.txt`

3. Make sure you compile and run the COBOL program without any problem with OpenCOBOL 1.1.0 and gcc 2.95.2 on Windows computers in SHB924/904. We will grade your works based on those machines.

4. The report should be submitted to VeriGuide, which will generate a submission receipt. The report and receipt should be submitted together with your COBOL and C codes in the same ZIP archive.

5. The COBOL source should have the filename "ta_ranking.cob".The C source should have the filename "ta_ranking.c". The report should have the filename "report.pdf". The VeriGuide receipt of report should have the filename "receipt.pdf". All file naming should be followed strictly and without the quotes.

6. `Tar` your source files to `username.tar` by

   ```
   tar cvf tmchan.tar ta_ranking.cob ta_ranking.c report.pdf receipt.pdf
   ```

7. `Gzip` the `tarred` file to `username.tar.gz` by

   ```
   gzip tmchan.tar
   ```

8. `Uuencode` the `gzipped` file and send it to the course account with the email title "HW1 *studentID yourName*" by

   ```
   uuencode tmchan.tar.gz tmchan.tar.gz \
   | mailx -s "HW1 1155234567 Chan Tai Man" csci3180@cse.cuhk.edu.hk
   ```

9. Please submit your assignment using your Unix accounts.

10. An acknowledgement email will be sent to you if your assignment is received. **DO NOT** delete or modify the acknowledgement email. You should contact your TAs for help if you do not receive the acknowledgement email within 5 minutes after your submission. **DO NOT** re-submit just because you do not receive the acknowledgement email.

11. You can check your submission status at

    `http://course.cse.cuhk.edu.hk/~csci3180/submit/hw1.html.`

12. You can re-submit your assignment, but we will only grade the latest submission.

13. Enjoy your work :>