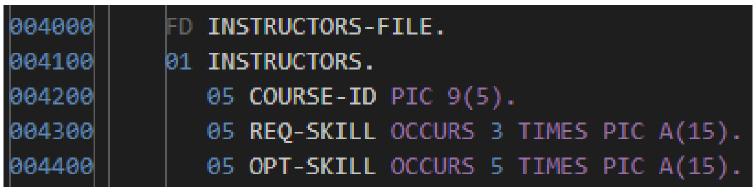
1. In reading files in a certain format, COBOL requires the spaces taken up for each variable in each line. We need to know exactly how many spaces used for each variable. And then COBOL can read the file and assign the data with a single line. So, it is very convenient to read the files in this exercise which have a fixed outline. While in C, it does not take special care for space used for the data. I need to take special care of the space within a string by changing the reading method to the space based.

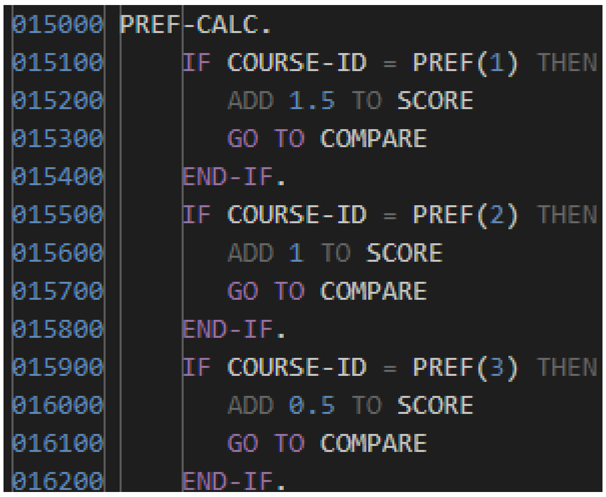


```
004000     FD INSTRUCTORS-FILE.
004100     01 INSTRUCTORS.
004200        05 COURSE-ID PIC 9(5).
004300        05 REQ-SKILL OCCURS 3 TIMES PIC A(15).
004400        05 OPT-SKILL OCCURS 5 TIMES PIC A(15).
```

Fig.1. File structure to read in COBOL

In variable use, COBOL only allows global variables while C allows both global and local variables. Using local variables is easier to debug as the scope limits its modification range. It is also more convenient as it allows variable name reuse.

In simulating loops, we are only allowed to use GO TO PARAGRAPH to simulate loops which is hard to follow. I need to keep tracking the start and the end of the loop. In compare with C, C is less likely to make and easier to fix the error since a loop finishes and returns in the same line and so the flow Is much clear in C.



```
015000 PREF-CALC.
015100     IF COURSE-ID = PREF(1) THEN
015200        ADD 1.5 TO SCORE
015300        GO TO COMPARE
015400     END-IF.
015500     IF COURSE-ID = PREF(2) THEN
015600        ADD 1 TO SCORE
015700        GO TO COMPARE
015800     END-IF.
015900     IF COURSE-ID = PREF(3) THEN
016000        ADD 0.5 TO SCORE
016100        GO TO COMPARE
016200     END-IF.
```

Fig.2. Looping in COBOL

In the function call, it is simulated by the paragraph in COBOL. It does not allow returning to the origin line. A new paragraph is needed to go back. It also cannot perform parameters passing because of the limitation of the variable using. Therefore, C is more convenience then COBOL.

2. COBOL is imperative programming paradigm while many modern programming languages (like Java) are multi-paradigm (compose of imperative, object-oriented…). Many modern programming languages allow programmers to declare variables anywhere, but COBOL does not. COBOL only have global variables, but most of the modern programming language also allows local variable which is good for memory allocation and perform polymorphism like Java. Many modern programming languages allow calls of function and return parameter, but the simulation of the function call in COBOL does not have a return statement.

3. I think it is not suitable for writing this kind of application, though it has serval benefits. COBOL does not allow dynamic memory allocation which requires reading files multiple time and make the programme harder to code and less efficient. However, it can force the programmer to save memory. Lack of function call makes the program very long and messy. It is hard to read and debug.

4. It is separated into three submodules.
   a. Read instructors file
   b. Read candidates file -> Check required skills -> Check optional skills -> Check preferences -> Ranking
      It loops until the end of the candidates file.
      It goes back to Read candidates file after Ranking (or after Check required skills if the candidate does not satisfy the requirement).
   c. Print output file (the rankings)
      The three steps above loops until the end of the instructors file.