1.

For the reading of file in certain format in COBOL, I only need to outline the file format and COBOL can read every row tuples from the file and assigning the data to the right variable as stated in the outline. In C, I have to think to read by word or by line and even if I read by line like COBOL, I have to assign the right data into the right variables by myself. The loop to get the data into the right variable in my C is eliminated in COBOL.

```c
for(j=1;j<9;j++){ //list[1:8]  =
    ter=fgets(list[j], 16, fp);
}
```

Fig.1. File Reading in C

For looping in COBOL, as this assignment only allows the use of 'GO TO', it is more difficult to implement in COBOL than C. Though the looping by 'GO TO' is similar to the the normal 'for loop' in C (the {} in C is replaced by 'GO TO' and paragraph name in COBOL, C is more convenient to use and less likely to make a bug, as programmar don't need to search back for the name of the paragraph and type it again after 'GO TO'.

```
COMPUTATION-PREF-1.
    IF I-COUNT = 0 THEN
        MOVE 1 TO I-COUNT
        MOVE 0 TO PLACE
        GO TO CALC_TOP3
    END-IF.
    compare and increment in score
    IF COURSE = PREF(J-COUNT) THEN
        MULTIPLY 0.5 BY I-COUNT GIVING TMP
        ADD TMP TO SCORE
        MOVE 1 TO I-COUNT
        MOVE 0 TO PLACE
        GO TO CALC_TOP3
    END-IF.
    SUBTRACT 1 FROM I-COUNT.
    ADD 1 TO J-COUNT.
    GO TO COMPUTATION-PREF-1.
```

Fig.2. Looping in COBOL (Code Segment of Preference Score Calculation)

```c
for(j=3;j>0;j--){ //preference score calc
    ter=fgets(buff, 6, fp1);//preference
    buff[4]='\0';
    if(strcmp(list[0],buff)==0 && zeroscore==0)
        tmp_score=tmp_score+0.5*j;
}
```

Fig.3. Looping in C (Code Segment of Preference Score Calculation)

For the function call, as PERFORM is not allowed in this assignment and the non-existent of local variable in COBOL, C is much more convenient to use than COBOL. This is because 'GO TO' does not allow the function return to the caller position which may make the code more spaghetti and encourage bad programming style. Moreover, the non-existent of local variable, no parameters passing through function call as they are all global variables and I need to take care of the value of all variables in different functions(paragraphs) carefully.

For the variable use, as COBOL only have global variables, it is difficult to perform dynamic memory allocation and creating structure like link-list. Therefore, to perform the some implementation structure in C and COBOL, I need to use array for the calculation of the top3 ranking instead of using link-list. Moreover, the flexibility of programming is reduced (like no recursion can be perform in a function) without local variables.

2.

For variable declarations, the abstraction in the modern programming languages is better than COBOL as people have to think about the format of type of variable declared (e.g. 9V9 for 1 digit with 1 significant figure) but people don't need to think about how many digit or characters they want for a variable in modern language like python. However, on the other view point, COBOL is more flexible for variable declarations.

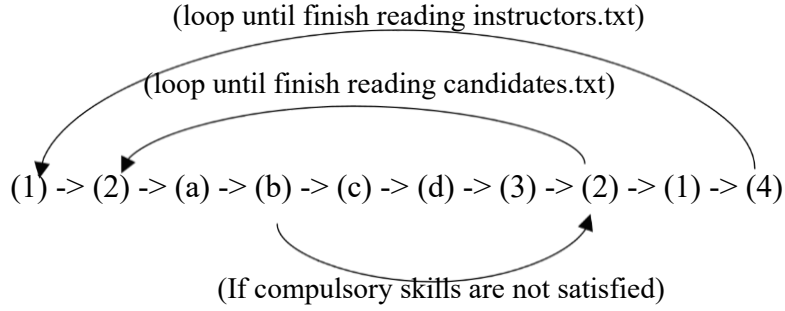
For paradigm, COBOL is an imperative programming language while many modern programming language like python is multi-paradigm language (compose of imperative, object-oriented, functional paradigm and so on).
For data type, COBOL only have global variables and thus it is difficult to perform dynamic memory allocation and creating structure like link-list comparing with the model languages like python.
For parameter passing, which is to pass parameters to a procedure or a function (usually through passing by reference or passing by value), COBOL cannot perform this because of the absence of local variable and the access to a data cannot be restricted unlike many modern programming language.
For writability and readability, COBOL is lacking structure because of the constant apply of 'GO TO' which allow the programme jump to and return from any procedure. Programmer can easily having bugs by coding and jumping too messy.
For reliability, because of the previously mentioned characteristic, this can cause of breaking of single entry single exist, increasing the difficulty of testing the reliability of the program.

3.
I think it is kind of suitable to write applications like this assignment in COBOL, which is more like reading data from record and processing them, because of the convenience of file reading and the scale of application. As the scale of these application is small which COBOL are capable to handle with (for large application, the lack of modularization of COBOL make it unsuitable to code). Thus, though the programming difficulty of coding maybe higher because of the inconvenience of function call, variable declaration and looping simulation, the scale of the application and the convenience in file reading make it acceptable. In addition, as COBOL is a compiled language which can gives a good performance in program efficiency than interpreted language like python.

4.
There are 4 submodules,

1) '**read instructors**': read the instructors.txt and pass the data to 'compute'
2) '**compute**': take the value of every tuple of instructor.txt with the data in candidate.txt and calculate the score of every candidates and pass to 'calc_top3' ,
    a. '**read candidates**': read the candidates.txt for the later comparison of data
    b. '**compute_complusory**': computation of compulsory skills of a course by comparing the 3 compulsory skills of a course and all skills that the candidate have; if the compulsory skills are not fulfilled, pass this candidate and calculate the score for the next candidate,
    c. '**compute_optional**': computation of the score for optional skills by comparison of the 5 optional skills a course and all skills that the candidate have,
    d. '**compute preference**': computation of the preference score by comparison of the course id and the preference list of the candidate,
3) '**calc_top3**': update the top3 ranking after intake the score or every candidates.
4) '**write file**': write the output.txt according to the top3 ranking

Flow of program:



Verbal Description of Program Flow:

The program first intake the data from instructors.txt and then pass every tuple to the submodule ‘compute’.
The top3 ranking is initialized with name as ‘0000000000’ and score as 0 for every intake of a tuple of data in instructors.txt.
Then, the submodule ‘compute’ read the data from candidates.txt and perform calculation in score by comparison between a tuple of instructors.txt and every tuple in candidates.txt.
The calculation procedure has 3 states (2b, c and d submodules).
Then the ranking is updated accordingly if the score calculated is higher than the candidate in the ranking in submodule ‘calc_top3’.
If the compulsory skills are not satisfied, jump to read another tuple of candidates and compare until the end of candidates.txt.
Finish reading and then write a tuple of a course in output.txt.
Repeat the process until finishing reading instructors.txt.