CSCI3180 Assignment 1 Report SID: 1155077469 Name: Choi Man Kin Email: mkchoi6@cse.cuhk.edu.hk

1. Conveniences and difficulties
    a. Reading file in certain format
    It is convenient to read file in certain format in both Cobol and Fortran. We can define the file structure or format, and it will automatically store the data in the defined variables. But Fortran is more flexible, we can define the format near the read statement, we can also pass different storing variable in read statements, even using the same format. We don't need to link the file format and file, when the program starts.
    b. Simulating loops/nested loops
    Simulating loop in Fortran is easier, because we can define the label or line for GO TO. We can still follow the modern structure like normal loop, and nested loop with break. But in Cobol, I have to define different procedures to make a block for the loop. Therefore, I need to design many procedures in Cobol. It is inconvenient to trace each variable and manage them inside many different procedures.
    c. Case control
    When I want to change some condition, I have to change the counter condition. It is inconvenient without else. If I was with else, I can only modify the condition in the if to achieve the change, line 325-333.
    d. Variable Scope
    There is no local variable in Cobol. I need to come out with many different names for each variable. This is quite headache. In contrast, Fortran has local variables. It is easier to manage the scope. I don't need to take care all procedures because of linked variables. The structure in Fortran is clearer.
2. Compare with modern programming languages, C#
    a. Paradigm
    C#, Fortran 77 and Cobol are imperative languages, but C# is a object-oriented language.
    b. Data type
    Fortran 77 and Cobol only have primitive data type, but C# have reference data type for string, and have object. C# supports both static and dynamic data type. C# does not support global variable and function. C# supports generic.
    c. Parameter parsing
    Cobol doesn't have parameter passing, only have global variable to pass data. Fortran has parameter. I can manage the scope of a variable. The main difference between Fortran and C# in parameter passing is that C# supports both value passing and reference passing depending the types of parameters. For example, integer is always passed by value.
3. Program design
    a. Submodules in Cobol

MAIN-Group:	MAIN-PROC, CREATE-OUTPUT-PROC, RESET-ALL-VARIABLES-PROC, END-PROC
CORE-Group:	RESET-TEAM-VARIABLES-PROC, TEAM-PROC, SCAN-RECORDS-LOOP-PROC, SCAN-RECORDS-ACTION-PROC, CAL-PROBLEM-SCORE-LOOP-PROC
DISPLAY-Group:	DISPLAY-HEADER-PROC, DISPLAY-TEAM-NAME-PROC, DISPLAY-PROBLEM-ID-PROC, DISPLAY-PROBLEM-SCORE-PROC, DISPLAY-TEAM-SCORE-PROC, DISPLAY-NEW-LINE-PROC

    b. MAIN-Group is just calling the CORE-Group. CORE-Group is calling DISPLAY-Group.
    CORE-Group formed the business logic. The inputs will be processed in this group. TEAM-PROC for teams.txt, SCAN-RECORDS-LOOP-PROC for submission-records.txt. After SCAN-RECORDS-LOOP-PROC see a different-team submission records, CAL-PROBLEM-SCORE-LOOP-PROC will be performed to calculate the problems' score, according to the spec. Then, display using DISPLAY-Group. Go back to TEAM-PROC for the next team and reset using RESET-TEAM-VARIABLES-PROC. If TEAM-PROC knows no more team, then go to END-PROC.
    c. Submodules in Fortran
    1. main 2. processTeam 3. displayTeamResults 4. resetTeamVariables
    d. If processTeam check whether there is more team, yes -> get it and perform processTeam, no -> end the program. If processTeam check whether it is the next team record, yes -> displayTeamResults and then go back to processTeam, no -> go to processTeam. The Fortran version has fewer submodules, because it supports labeling go to. I don't need to extract the looping block in an individual submodule.