Name: Kwan Ting Fung ID: 1155110979 Year: 3

Assignment 1 - TA ranking system in COBOL and C - Report

1. To compare the conveniences and difficulties in implementing the system, I divide the process into tasks.

   In terms of “reading file in certain format”, I think COBOL is more convenient than C. Because COBOL could store the format of a file and save to global variable easily, which doesn’t need to care about pointer stuff or local variable so much.

```
i.e. COBOL:                                  C:
        FILE SECTION.                         file* f_c_ptr;
        FD. CANDIDATES.                      f_c_ptr = fopen(“candidates.txt”, “r”);
        01 CANDIDATES-FILE.                  struct candidates{
        05 C-ID PIC X(11).                       int iD[11];
        05 …….                                              ...
                                             };
        WORKING-STORAGE SECTION.                 struct candidates input;
        01 WS-CANDIDATES.                        fgets(input.iD, 11, f_c_ptr);
                05 WS-C-ID PIC X(11).            fgets(...);
                05 …...
        READ CANDIDATES INTO CANDIDATES-FILE
        MOVE CANDIDATES-FILE TO WS-CANDIDATES
```

   But **if** it is “reading in variance format of different lines”, C should be better as it has different reading functions and controllable parameter that improves flexibility, while COBOL only has 3 reading mode, and fixed the format in global variable.

   In terms of “simulating loops”, obviously C is better as there is “for” and “while” functions, while COBOL has “GO TO” only (for this assignment), “GO TO” is too easy to use that considering harmful. Programmers will make spaghetti code if they don’t think out the consequence of “GO TO” , like using goto for quiting a loop, it is 99% dangerous. This is only good for exception handling. C is more convenient as “for” and “while” are “single input, single output”.
   C has higher writability and expressivity, thus more convenient to programmer.

```
i.e. COBOL:                                       C:
        LOOP-PARA.                                 for(int i=0; i<5;i++){
                DISPLAY “Hello!”                          printf(“Hello!”);
                ADD 1 TO WS-INDICE                 }
                IF WS-INDICE <=5 THEN                            or
                        GO TO LOOP-PARA            int i=0; while(i<5){
                END-IF.                                 printf(“Hello!”); i++}
      *if there is something under this loop, probably becomes spaghetti.
```

#COBOL displays 5 “Hello!” in seperate lines, while C displays in one single line.

In terms of “function call”, obviously C functions have parameters, while COBOL calls paragraphs, or call other COBOL programs only, but no parameters. I think it is fair for convenient or difficulties. C supports for abstraction, programmers could define and use complex structures that allows details to be ignored. It could be convenient. But it has a larger number of primitives and higher orthogonality that may lead to some undetected errors. COBOL has smaller number of primitives and has a consistent set of rules for combination, bugs could be easily detected,. But relatively, it doesn’t provides a much more complex structures.

In terms of “variable declaration”, C is more convenient as C allows local variable, such that we could have identical names but different usage as they are in local and global variable separately. But COBOL only has global variable, which we need to take care of it really carefully.

2. COBOL is imperative and designed for business, which was created before appearance of internet, so it is not good for internet development. It is good for OO-base abstraction. It provides global variable only, such that all variable names could not be the same(case-sensitive). It doesn’t have any functions, so no parameter parsing (except for linkage section), so COBOL could not support for functional programming. COBOL is not strongly-typed, it has only three data types, numeric, alphanumeric, and alphabetic. They are weakly enforced by compiler, thus, it is possible to assign a non-numeric value to a numeric data item. Hence, COBOL programmers should make sure that non-numeric data is never assigned to numeric items for calculation. It is not for declarative programming. And it is needed to declare the number of “bit” for each variables, i.e. PIC X(9).

   C++ is imperative and designed for efficiency. It could be multi-paradigm : procedural , functional, object-oriented, and generic. It could have global and local variables, it allows polymorphism and have different group of data types,i.e. character, numerical integer, floating-point, boolean, void, null. These data types’s size of bit are well-defined, i.e. singed int for 16 bits. So, every functions have a return type, and pass different type of parameters, which could be in value, in pointer, or in reference.

3. I think it is not suitable for this assignment. Because it is only allowed to use if and goto, such that the loops are simulated by paragraph recursion, which is not efficient.. Some of the input needs to be passed several times, so many indices are needed to be created, however, COBOL only has global variable. Thus, we need to handle the indice name carefully and without crash, it causes troubles in writability. To conclude, apparently, it is not good for sorting. But, somehow it is good for reading files in

certain format. That means, it is ok for data transaction and calculation, just like normal business, but not good for writing applications and output files.

4. I seperate the tasks into many submodules: open file, exception, length, store, valid, optional, preference, manager, swap, bubble sort, output.

    1. “Open file” means open file literally, if files are not exist, it passes to exception.
    2. “Exception” means exception handling, quit program at once and display not exist. It also handles empty files for saving times.
    3. “Length” is to calculate the length of “entry” in a file, i.e. 4 courses, 6 TAs.
    4. “Store” is to use the calculated length in “Length”, to initialize the table (COBOL) or structure array (C), then open the files again, and save those data into those data structures. If the length is equal to zero, empty file exception is directed.
    5. “Valid” is to calculate if one specific TA is valid for one specific course.
    6. “Option” is to calculate the score of one specific TA if his skills are matched with optional skills for one specific course.
    7. “Preference” is to calculate the score of one specific TA, which his preference is matched with the specific course ID.
    8. “Manager” is to perform a manager role, call the “Valid”, “Option”, “Preference” for each courses and each TAs, and save the results to another data structure or table, i.e. table{TAID, Scores}.
    9. “Swap” is to swap specfic two account of the above table data structure.
    10. “Bubble Sort”, using “Swap” to perform bubble sort in the table made by “Manager”.
    11. “Output”, extract the first three account in the table after “Bubble Sort”, and insert them into output file.

My program main flow is:

```
File Exception(){
        non-exist file => exit(0) / STOP RUN
}
open file{
        if not exist, File exception
}
length{
        read,  count the dynamic length
}
store{
        read file,  store to particular structrures.
        if length ==0, empty exception
}
Empty Exception(flat){
        flag == 0, empty instructor => empty output
        flag == 1, empty candidates => extract ID from courses => output
}
Manager{
        for each course{
                for each TA{
                        if Valid, score = Option + Preference
                        else, score = 0
                }
                (COBOL) save to a temp table
                (COBOL) Bubble Sort
                (COBOL) Output
        }
        (C) save to particular array
}
```

I accidently implement two different managers in C and Cobol.

```
BubbleSort{
        for(i<length){
                if (score[i]<=score[j]) , Swap
                i++,j++;
                check for no more swaps
        }
}

Output{
        take the first three account in (COBOL) temp table / (C) array,
        write to output.txt
}
```