Name: Alex Lai
Student Number: 0920158
Email: alai02@uoguelph.ca

CIS*3190 - A1 Reflection Report - Fortran Tic-Tac-Toe

This is a report regarding the A1 assignment for CIS*3190. The assignment is called Fortran: playing tic-tac-toe. The program starts of by showing the user a board and instructing them to select a number from 1 - 9 and then showing them the first box that is filled with a X marking their move. The program will then place the computers move in an empty square and the turn is back on the user. The game continues until there is 3 characters in a line that are the same symbol or there are no more possible moves and the game results in a draw. The computer uses a weak AI algorithm to calculate its move and will win if there is a possible box in the board that will result in the computer winning. If there no winning moves the computer will then place their character in a spot that would block the user from directly winning the game. If neither of those 2 scenarios are present, the computer will place their character in a random square on the board that is not occupied. I used the fortran 95 standard in my implementation of the program.

To compile the program, type the following command: gfortran -Wall a1.f95
To run the program, type the following command: ./a.out

When designing the program, the first step that I took was talking the legacy code and modernizing the styling and syntax. The f77 standard used the C character as the comment symbol and the f95 standard uses the ! character. This is the first change that I made. The second change that I made was switching all the code to use lower case letters because letter casing does not matter for the f95 standard. The third step that I took in designing this program was implementing the helper functions of the program (same, pickMove, showBoard, ect…). Next I changed the variable declarations, if statements, and loops to the syntax of the f95 standard for the legacy functions. After that I began testing and debugging the program until I got a simple working tic-tac-toe program. I looked for places I could optimize the program and places I can simplify the code. I repeated the process of testing and implementing until I decided there were no more optimizations for this program. Next I added error checking for the users input to make sure the game does not crash when an invalid move is played (ex. The user enters a character when they need to enter an integer). Finally, I commented my code and cleaned up the styling for a finished program.

The largest problem with re-engineering this code and writing the fortran program was debugging the fortran specific bugs that I have no encountered in other languages. For example, when using a boolean to check for a valid computer move I was initializing the variable at the top of the subroutine and using it later on in the subroutine. What I did not know was that when calling the subroutine, it would not reinitialize the variable and it would be in the state it was in from the last call to that subroutine. This was something that would not have occured in a language like C and it took me a while to figure out where the bug was. Another idiosyncrasy of

fortran was the fact that logical operators is different from comparing characters and integers. I kept using regular operators that I have been used to just out of habit and would have to go back and change the logical operators to the dot notation that it uses.

The simplicity of setting a parameter variable passed into a subroutine was something I found very useful in fortran. Just having to declare intent in/out and not needing to worry about where the memory for the variable is made using subroutines a much more favorable option compared to functions which could only return a single value. This was also very useful because unlike C, it does not require the programmer to handle the memory that is passed in and out of subroutines. Also the fact that functions require you to make the return value equal to the function name made for less variable declarations in smaller functions that did not require the return value to change or be compared to in the function. Fortran definitely has some great features and styling to write great reliable and readable code for a given task and will always have problems that would be great for a solution in the language. It is not great for larger complex programs that require large data structures and more of an object oriented design approach since there are much optimized for that style of coding but I don't see fortran programs being completely rewritten in other languages any time soon unless requirements or other changes are a factor. I think fortran would have been a much harder language to learn if it was someone's first language because it is not very user friendly compared to something like Java and it would be a steep learning curve because of things like figuring out how types, loops, and functions/subroutines work. Having experience in another language gave me a ton of help since I was able to reference all my programming knowledge and just find out how to implement it differently.

I think the program would have been easier to write in C because of the amount of programs I have written in C compared to having to learn the fortran language and having to implement the solution in a language I was very new to. Aside from the initial learning curve, I think the program would have been very similar to a implementation in C but maybe would have been shorter because of the amount of libraries C has at its disposal that I think could be used to optimize the program. I think the core logic wouldn't vary too much and the data types used would be very similar. Overall I was pleased with the program I wrote and found fortran a unique language to learn and definitely have a new appreciation for legacy languages which I did not have prior.

Fortran was overall an easy language to learn because of its similarities to C. I found each new concept learned about the fortran language had a similar concept or implementation in C. The hardest part about learning fortran was trying to find the proper syntax to use when writing a piece of code. For example, checking for bad user input took longer than expected since I had to find out how the read function uses parameters to get proper user input. Also finding a valid solution to the random number generator took a bit of searching since it required a function that I was not familiar with.