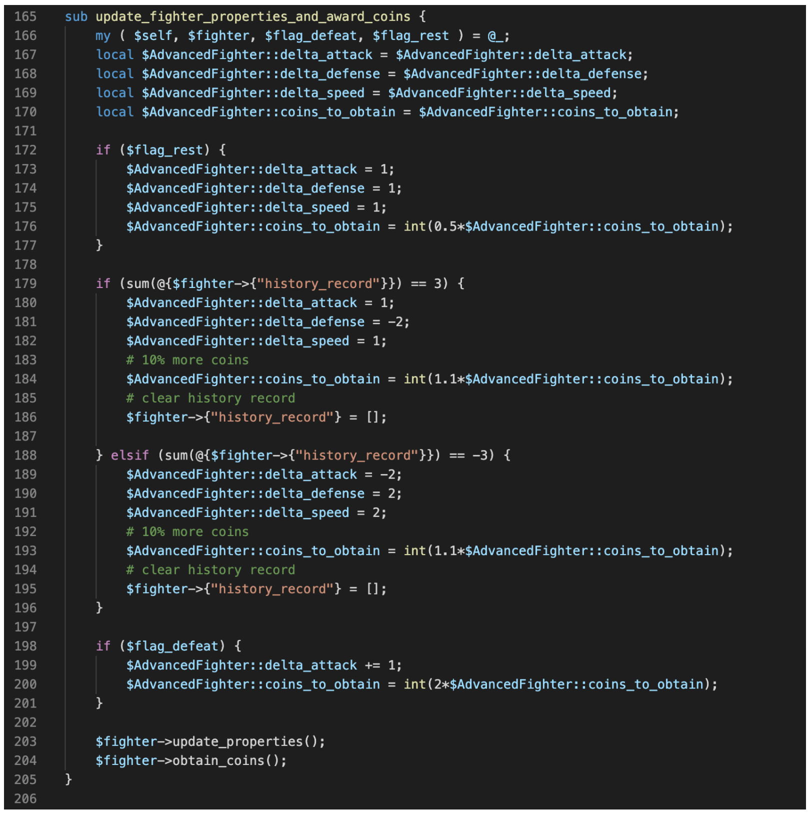
<u>CSCI3180 Assignment 3 Report</u>

Nikunj Taneja (SID: 1155123371)

12 Apr 2022

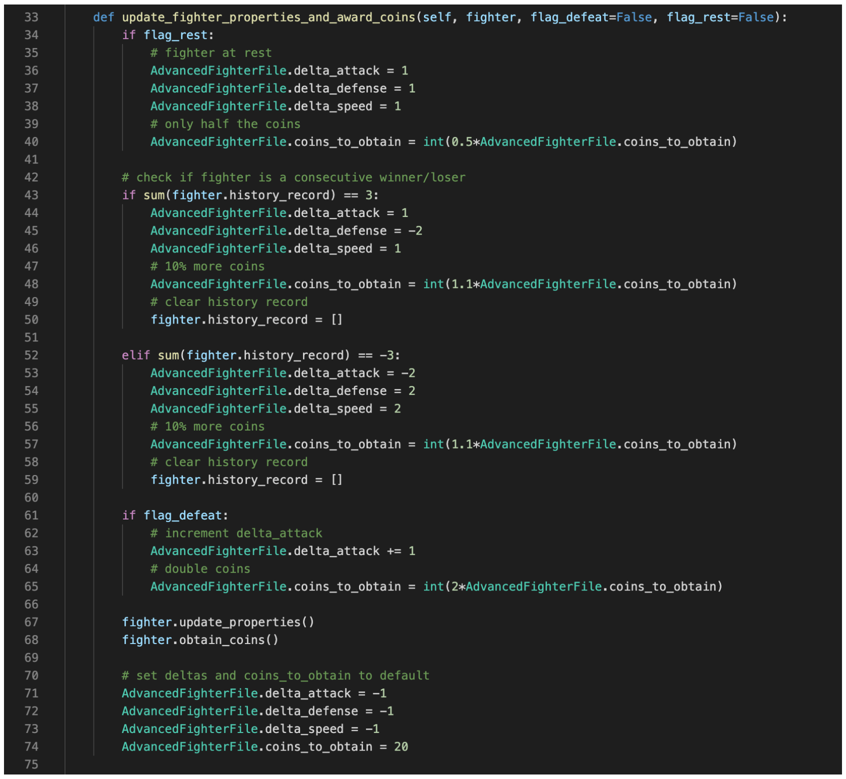
1. Provide example code and necessary elaborations for demonstrating the advantages of Dynamic Scoping in using Perl to implement the Advanced Tournament Duel game as compared to the corresponding codes in Python.

The main advantage of using dynamic scoping to implement the Advanced Tournament Duel game is the fact that there is no need to reset the global variables back to default values after *temporarily* changing them. Since all the changes made to *delta_attack, delta_defense, delta_speed* and *coins_to_obtain* are <u>temporary</u> in nature, we don't need to worry about the side effects of changing the mentioned global variables since only the *local copies* of those variables contain those changes. This is illustrated in Figure 1.



```perl
sub update_fighter_properties_and_award_coins {
    my ( $self, $fighter, $flag_defeat, $flag_rest ) = @_;
    local $AdvancedFighter::delta_attack = $AdvancedFighter::delta_attack;
    local $AdvancedFighter::delta_defense = $AdvancedFighter::delta_defense;
    local $AdvancedFighter::delta_speed = $AdvancedFighter::delta_speed;
    local $AdvancedFighter::coins_to_obtain = $AdvancedFighter::coins_to_obtain;

    if ($flag_rest) {
        $AdvancedFighter::delta_attack = 1;
        $AdvancedFighter::delta_defense = 1;
        $AdvancedFighter::delta_speed = 1;
        $AdvancedFighter::coins_to_obtain = int(0.5*$AdvancedFighter::coins_to_obtain);
    }

    if (sum(@{$fighter->{"history_record"}}) == 3) {
        $AdvancedFighter::delta_attack = 1;
        $AdvancedFighter::delta_defense = -2;
        $AdvancedFighter::delta_speed = 1;
        # 10% more coins
        $AdvancedFighter::coins_to_obtain = int(1.1*$AdvancedFighter::coins_to_obtain);
        # clear history record
        $fighter->{"history_record"} = [];

    } elsif (sum(@{$fighter->{"history_record"}}) == -3) {
        $AdvancedFighter::delta_attack = -2;
        $AdvancedFighter::delta_defense = 2;
        $AdvancedFighter::delta_speed = 2;
        # 10% more coins
        $AdvancedFighter::coins_to_obtain = int(1.1*$AdvancedFighter::coins_to_obtain);
        # clear history record
        $fighter->{"history_record"} = [];
    }

    if ($flag_defeat) {
        $AdvancedFighter::delta_attack += 1;
        $AdvancedFighter::coins_to_obtain = int(2*$AdvancedFighter::coins_to_obtain);
    }

    $fighter->update_properties();
    $fighter->obtain_coins();
}

```

Figure 1a. Implementation of Advanced Tournament with Dynamic Scoping in Perl



```python
33      def update_fighter_properties_and_award_coins(self, fighter, flag_defeat=False, flag_rest=False):
34          if flag_rest:
35              # fighter at rest
36              AdvancedFighterFile.delta_attack = 1
37              AdvancedFighterFile.delta_defense = 1
38              AdvancedFighterFile.delta_speed = 1
39              # only half the coins
40              AdvancedFighterFile.coins_to_obtain = int(0.5*AdvancedFighterFile.coins_to_obtain)
41
42          # check if fighter is a consecutive winner/loser
43          if sum(fighter.history_record) == 3:
44              AdvancedFighterFile.delta_attack = 1
45              AdvancedFighterFile.delta_defense = -2
46              AdvancedFighterFile.delta_speed = 1
47              # 10% more coins
48              AdvancedFighterFile.coins_to_obtain = int(1.1*AdvancedFighterFile.coins_to_obtain)
49              # clear history record
50              fighter.history_record = []
51
52          elif sum(fighter.history_record) == -3:
53              AdvancedFighterFile.delta_attack = -2
54              AdvancedFighterFile.delta_defense = 2
55              AdvancedFighterFile.delta_speed = 2
56              # 10% more coins
57              AdvancedFighterFile.coins_to_obtain = int(1.1*AdvancedFighterFile.coins_to_obtain)
58              # clear history record
59              fighter.history_record = []
60
61          if flag_defeat:
62              # increment delta_attack
63              AdvancedFighterFile.delta_attack += 1
64              # double coins
65              AdvancedFighterFile.coins_to_obtain = int(2*AdvancedFighterFile.coins_to_obtain)
66
67          fighter.update_properties()
68          fighter.obtain_coins()
69
70          # set deltas and coins_to_obtain to default
71          AdvancedFighterFile.delta_attack = -1
72          AdvancedFighterFile.delta_defense = -1
73          AdvancedFighterFile.delta_speed = -1
74          AdvancedFighterFile.coins_to_obtain = 20
75
```

Figure 1b. Implementation of Advanced Tournament without Dynamic Scoping in Python

Note the fact that the variables have to be set back to default values in the Python code (lines 71-74), while there is no need to do this in Perl. Assuming the programmer and the person reading the code is familiar with the concept of dynamic scoping in Perl, this increases the readability of the code.

2. Discuss the keyword local in Perl (e.g. its origin, its role in Perl, and real practical applications of it) and give your own opinions.

The *local* keyword in Perl gives a temporary, dynamically-scoped value to a global (package) variable, which lasts until the end of the enclosing block. It also enables the functions called within that enclosing block to access that value. It helps contain all the changes made to an otherwise global variable to the current scope, i.e., the enclosing block that declares it, all the functions called in that block, all the functions called in those functions, and so on. This is possible because once the enclosing block is finished, the original global value is restored to that variable. This especially comes in handy when dealing with localized changes, e.g., in localized filehandles and local aliases [1]. It can even help eliminate the need for passing variables to functions, as can be seen in Figure 1a. In my opinion, if used correctly, the *local* keyword is a nice feature to have but even without it, I could live happily ever after.