Cardiff School of Computer Science and Informatics

Coursework Assessment Pro-forma

Module Code: CM6114

Module Title: Computational Thinking

Lecturer: Alexia Zoumpoulaki

Assessment Title: Programming Assignment

Assessment Number: 001

Date Set: 19/10/2018

Submission Date and Time: 16/11/2018 at 9:30am.

Return Date: 7/12/2018

Table of Contents

Documentation

Describe your implementation

Problem Description

The task give was to create a system/programme for a simulation of a tennis tournament. In the programme/system the task given was to select the top 12 players based on their ranking across Wales out of the list of other players. They must be initially randomly assigned in groups of three, totalling into four groups. Where they play mini round-robin tournaments (or all-play-all tournament). That means that each plyer will meet all other players in turn. Afterwards the top players of each group (if there is a tie, meaning all players have won 1 game, then the player with the highest-ranking progresses) play an elimination tournament. The 4 players are assigned to 2 matches randomly and the winners from the matches play to the final.

Problem Decomposition

1. Open tennis tournament file and store it in a array/dictionary
2. Sort players in ascending order of ranks
3. Select top 12 players
4. Assign each player a strength according to rank e.g. rank 1 should have highest strength
5. Randomly assign 3 players to a group
6. Setup the matchups between players for each group
7. Do coin flip for each serving in the game, whoever wins should output their score and the other player's score
8. Add points to a player for each serving with probability
9. Resolve any ties by having a player win 2pts in a row
10. First Player who wins 6 games wins a set
11. Player who wins 2 sets wins the match, 3 sets if they both win 1 set each
12. Record the scores throughout the game
13. Take winners of each group and assign them into random groups of two
14. They should all play each other and the two winners of this round will play final round
15. Select winner of the final round

Pseudo code

#Variables

```
Set top_twelve_players as array
Set players as array
Set group_1 as dictionary
Set group_2 as dictionary
Set group_3 as dictionary
Set group_4 as dictionary
Setlevel_1 as dictionary
Setlevel_1_scores as dictionary
Set game_scoring as dictionary = {
   0: 'love',
   1: '15',
   2: '30',
   3: '40',
   4: 'Game'
}
```

#opening file

```
function open file
   open tournament_players.csv file:
      rows = file
      for each row
         add to players read row 1 to 3
     endfor
   for i to the end of length of players do
      if a players ranking is empty
         player rank = random number between 1 to 18
      endif
   endfor
   for i to end of length of players do
      players ranking = player ranking as integer
  endfor
endfunction
```

selecting top 12 players according to ranking putting into list variable called top_twelve_players

```
function top_twelve_players_sort
                rankings = sort ranking in ascending order
                for i  from 0 to 12 do
                   add top 12 players to top_twelve_players
               endfor
endfunction
```

```
Call open file
Call top_twelve_players_sort
```

#Player strengths

```
player_strengths = copy of top_twelve_players
j = 0
player_strength_points = length of player_strengths + 28
for i to (length of player_strengths) do
   player_strength_points  = player_strength_points - 3
   add (top_twelve_players ranking + player_strength_points) to player_strengths
endfor
Output player_strengths
```

#assign random players to group

```
Randomize player_strengths
for n from 0 to end of player strengths,3 do
   for i to player_strengths[n:n+3] do
      add player name and player strength to level_1
   endfor
endfor
group_1, group_2, group_3, group_4 = level_1
Output level_1
```

#tournament matchups

```
function mini_robin_tournaments(groups)
   list2 = combine groups in pairs of 2's
   global variable group_number
   group_number = group_number 1
   Output group_number match-ups
   for i to end of list2 do
      contestants = i
      Output contestants0 vs contestants1
      call games and pass contestants0 and contestants1
   endfor
endfunction
```

#games

```
function games(p1, p2):
   call add_points and pass p1 and p2
endfunction
```

#points

```
function add_points(player1, player2):
   player_1_set = 0
   player_2_set = 0
   while player_2_set != 3 and player_1_set != 3:
      player_1_game = 0
      player_2_game = 0
      while player_1_game != 6 and player_2_game != 6:
         player_2_pt = 0
         player_1_pt = 0
         game_won = False
         while not game_won do
            call coin_flip and pass player_1_pt, player_2_pt, player1 and player2
            player1_prob = call search and pass in level_1, player1
            player2_prob = call search and pass in level_1, player2
            player1_prob = player1_prob + random number between 8 and  50
            player2_prob += player2_prob + random number between 8 and  50
            if player_1_pt > 3 and player_1_pt >= (player_2_pt+2) do
               Output  player1 Won the game
               player_2_pt = 0
               game_won = True
               player_1_game = player_1_game  + 1
            else if player_2_pt > 3 and player_2_pt >= player_1_pt + 2 do
               Output player2 Won the game
               player_1_pt = 0
               game_won = True
               player_2_game += 1
            endif
            if player_1_pt >= 3 and player_1_pt = player_2_pt do
               Output 40-all, deuce
            endif
            if player1_prob > player2_prob:
               player_1_pt = player_1_pt + 1
            else if player2_prob > player1_prob:
               player_2_pt = player_2_pt + 1
            endif
      endwhile
      if player_1_game = 6 do
         Output player1 won a set
         player_1_set = player_1_set + 1
         Output player_1_set '-' player_2_set
      else if player_2_game = 6:
         Output player2 won a set
         player_2_set = player_2_set + 1
         Output player_2_set '–' player_1_set
       endif
   endwhile
   if player_1_set = 3 do
      Output player1 won the match
      add player1 name and score to level_1_scores
   else if player_2_set = 3 do
      Output player2 won the match
      add player 2 name and score to level_1_scores
   endif
endfunction
```

#search for player strength

```
fucntion search with values, searchFor
   for j to end of values do
      for k to end of  j do
         if searchFor in k
           return j[k]
         endif
      endfor
   endfor
endfunction
```

#coin flip

```
fucntion coin_flip(player_1_pt, player_2_pt, player1, player2):
   coin = random of heads or tails
   if player_1_pt > 4 or player_2_pt > 4 do
      if coin = 'heads' do
         Output  Heads: player1 serves first
         Output player_1_pt '-' player_2_pt
      else:
         Output Tails:  player2  serves first
         Output player_2_pt '-' player_1_pt
      endif
   else:
      if coin = 'heads' do
         Output  Heads:  player1 serves first
         Output game_scoring(player_1_pt) '-' game_scoring(player_2_pt)
      else:
         Output Tails: player2 serves first
         Output game_scoring(player_2_pt) '–' game_scoring(player_1_pt)
      endif
  endif
endfunction
```

#code

```
group_number = 0
call mini_robin_tournaments and pass group_1
call mini_robin_tournaments and pass group_2
call mini_robin_tournaments and pass group_3
call mini_robin_tournaments and pass group_4
Output level_1_scores
```

Data representation

Variable	Type
`top_twelve_players = []`	List
game_scoring = { **0: 'love',** **1: '15',** **2: '30',** **3: '40',** **4: 'Game'** **}**	Dictionary
player_strength_points -= 3	Integer
Players = [['Summer Smith', 1, 38], ['Scary Terry', 2, 36], ['Morty Smith', 3, 34]]	Lists of list
def openfile():	Function
Leve_1 = [{'Alan Rails': [16, 12], 'Abradolf Lincler': [29, 4], 'Tophat Jones': [27, 5]}]	list of dictionary
contestants = I (player names)	String

Functions

```
# """selecting top 12 players according to ranking putting into list variable called top_twelve_players"""
def top_twelve_players_sort():
    rankings = sorted(players, key=lambda i: i[1])  # sorts players into chronological order in rankings
    for i in range(0, 12):
        top_twelve_players.append(rankings[i])  # picks out top 12 players
```

```
openfile()
top_twelve_players_sort()
player_strengths = top_twelve_players.copy()
assign_player_strength()
```

The purpose of the function above is to sort the players by ranking and picking out the top twelve. A call is made to top_twelve_players_sort() which then runs the code inside it. There is no argument required in this case as I simply want to just operate it so it picks out the top players and stores it in a list.

```python
# """assigns player strength and sorts player in player strength order"""
def assign_player_strength():
    j = 0
    player_strength_points = len(player_strengths) + 28  # creates player strength points
    for i in range(len(player_strengths)):
        player_strength_points -= 3
        player_strengths[i].append(
            top_twelve_players[i][1] + player_strength_points)  # assigns player strength according to ranking
    print(player_strengths)
```

```python
openfile()
top_twelve_players_sort()
player_strengths = top_twelve_players.copy()
assign_player_strength()
```

The assign_player_strength function simply will add player strength to player_strengths variable it will create player strength from ranking + 28 for each player and -3 as we go down the list and store it in the list as well.

Flow Control

```
    if player_1_pt > 3 and player_1_pt >= (player_2_pt+2):
        # uncomment code below to view messages
        # print('\n'f".......................")
        # print(f'\n{player1} Won the game')
        player_2_pt = 0
        game_won = True
        player_1_game += 1
    elif player_2_pt > 3 and player_2_pt >= (player_1_pt+2)
        # uncomment code below to view messages
        # print(f'\n{player2} Won the game')
        # print('\n'f".......................")
        player_1_pt = 0
        game_won = True
        player_2_game += 1
    #     uncomment the code below to view messages
    # if player_1_pt >= 3 and player_1_pt == player_2_pt:
        # print('40-all  deuce')
```

```
# """assigns player strength and sorts player in player strength order"""
def assign_player_strength():
    j = 0
    player_strength_points = len(player_strengths) + 28  # creates player strength points
    for i in range(len(player_strengths)):
        player_strength_points -= 3
        player_strengths[i].append(top_twelve_players[i][1] + player_strength_points)  # assigns player strength according to ranking
    print(player_strengths)
```

```
if player_1_set == 3:
    print(f'     {player1} won the match ')
    duplicate = False
    response = sort_out_duplicates(player1)
    if response is True:
        level_1_scores.append([player1, 2])
    else:
        level_1_scores.append([player1, 1])
elif player_2_set == 3:
    print(f'     {player2} won the match')
    duplicate = False
    response = sort_out_duplicates(player2)
    if response is True:
        level_1_scores.append([player2, 2])
    else:
        level_1_scores.append([player2, 1])
```

Examples of good practices

```python
# """assigns player strength and sorts player in player strength order"""
def assign_player_strength():
    j = 0
    player_strength_points = len(player_strengths) + 28  # creates player strength points
    for i in range(len(player_strengths)):
        player_strength_points -= 3
        player_strengths[i].append(top_twelve_players[i][1] + player_strength_points)  # assigns player strength according to ranking
    print(player_strengths)
```

```python
# """selecting top 12 players according to ranking putting into list variable called top_twelve_players"""
def top_twelve_players_sort():
    rankings = sorted(players, key=lambda i: i[1])  # sorts players into chronological order in rankings
    for i in range(0, 12):
        top_twelve_players.append(rankings[i])  # picks out top 12 players
```

Error Handling

```
result = False
while not result:
    try:
        user_input = int(input('Enter the number of simulations you want to run: '))
        result = True
    except ValueError as e:
        print('Must be only integers')
    else:
        if user_input < 50:
            print('Must be greater than 50')
            result = False
```

```
# Opening file and putting it into players
def openfile():
    try:
        with open(sys.argv[1], 'r') as f:
            rows = csv.reader(f)
            for row in rows:
                players.append(row[1:3])  # Only print the column in the row
            del players[0]
        for i in range(len(players)):
            if players[i][1] is '':  # finds any players without a rank
                players[i][1] = random.randint(1, 18)  # assigns a random rank to a player if the player has no rank
        for i in range(len(players)):
            players[i][1] = (int(players[i][1]))  # converts string into integer
    except OSError:
        print('cannot open')
    except IndexError:
        print('out of index')
    else:
        f.close()
```