Assurance Foundations Quiz 1

Alfred Murabito

February 8, 2020

Contents

Acknowledgments: I received no external help for this report.

Chapter 1

Executive Summary

All requirements for this project are satisfied. Specifically,

Report Contents
Our report has the following content:

- Chapter 1: Executive Summary
- Chapter 2: Report Body
  - Section 2.1: Problem statement
- Chapter 3 Code and Tests
  - Section 3.1: Relevant code
  - Section 3.2: Test results
- Chapter A: Source Code
- Chapter B: Addendum

Chapter 2

Report Body

2.1 Problem Statement

The Fleece'm'n-How law firm Generating Law has a need to keep a ratio of male to female workers above 50 percent as well as a combined total experience of at least 60,000 hours. Mr How has decided to analyze the total experience and gender ratio of the Junior Partners since he heard it is best if the firm is comprised of young workers.

1. To represent the table of data in ML, I used a (string * string * string * int) list data structure.
2. To calculate the total hours of all junior partners, I created a function that accepts the job status of interest and a list of employee entries with the important fields being job status, gender, and experience. It recursively iterates through each element in the list using constructors and pattern matching and adds an elements experience to the running total if the job status matches (job status, experience, and gender are extracted from the list entry using pattern matching).
3. To get the male/female split of a given job status,I used a similar approach as before to iterate through each entry in the list. If the entry meets the criteria for a male or female of the given job status, then the corresponding element in the tuple(male or female) will be incremented in the tuple returned using pattern matching.

Chapter 3

Special Coding and Tests

3.1 Relevant Code

3.1.1 Employee Firm Data

First piece of code we need is a definition in ML of the table containing all of the relevant employee information.

The following code is from *table.sml*

```
(* Table of relevant information in the form of
a (string * string * string * int) list


(*****************************************************************************)
(*    Table of relevant information in the form                               *)
(*    a (string * string * string * int) list                                 *)
(*                                                                            *)
(*****************************************************************************)
val table =
[("Nancy", "F", "Associate", 7),
("Peter", "M", "Associate", 5),
("Paul", "M", "Sr. Partner", 23),
("Robert", "M", "Jr. Partner", 11),
("Ralph", "M", "Associate", 4),
("Rhonda", "F", "Associate", 11),
("Sam", "M", "Associate", 5),
("Sara", "F", "Associate", 9),
("Alice", "F", "Sr. Partner", 27),
("Bob", "M", "Jr. Partner", 14),
("Eve", "F", "Associate", 9),
("John", "M", "Jr. Partner", 17),
("Gary", "M", "Associate", 6),
("Adam", "M", "Associate", 8),
("Carl", "M", "Associate", 2),
("Douglas", "M", "Jr. Partner", 14),
("Sally", "F", "Jr. Partner", 19),
("Joseph", "M", "Sr. Partner", 33),
("Mary", "F", "Jr. Partner", 18),
("Esther", "F", "Associate", 8),
("Frank", "M", "Associate", 7),
("Jane", "F", "Associate", 3),
("George", "M", "Jr. Partner", 15),
("Harry", "M", "Associate", 1),
("Robin", "F", "Associate", 8),
("David", "M", "Jr. Partner", 16),
```

```
("Jan", "F", "Associate", 6),
("Jack", "M", "Associate", 4),
("Ken", "M", "Associate", 9),
("Kathy", "F", "Associate", 1),
("Larry", "M", "Associate", 7),
("Mike", "M", "Jr. Partner", 12),
("Ted", "M", "Jr. Partner", 18),
("Terry", "M", "Sr. Partner", 30),
("Victor", "M", "Associate", 3),
("Vera", "F", "Sr. Partner", 28),
("Wilma", "F", "Associate", 7),
("Zack", "M", "Associate", 1),
("Debra", "F", "Associate", 12),
("Bonnie", "F", "Associate", 5),
("Dan", "M", "Jr. Partner", 17),
("Meredith", "M", "Jr. Partner", 17),
("Randal", "M", "Associate", 9),
("Jake", "M", "Jr. Partner", 12),
("Karen", "F", "Jr. Partner", 9)];
```

3.1.2 Calculate Total Hours Experience

```
(* Calculate the total hours in experience of  *)
(* a given seniority (status)                  *)

fun calc_h (status:string) [] = 0
  | calc_h (status) (head::table:(string*string*string*int)list) =
  let
    val (name, gender, j_status, exp) = head
    val match:bool = (status = j_status)
  in
    if match then (2080*exp)+(calc_h status table) else (calc_h status table)
  end;

(* For calculating total hours for all Jr. Partners, use below *)
calc_h ``Jr. Partner'' table;
```

3.1.3 Get Male/Female Split

```
(* Calculate the male/female split *)

fun get_male_female(status:string) [] = (0, 0)
  | get_male_female(status:string) (head::table) =
  let
    (* Use pattern matching to get elmenets of list entry *)
    val (name, gender, j_status, exp) = head
    val f_match:bool = ((status = j_status) andalso (gender = "F"))
    val m_match:bool = ((status = j_status) andalso (gender = "M"))
    val (p_male, p_fem) = get_male_female status table
  in
    ((if m_match then 1 else 0)+p_male,(if f_match then 1 else 0)+p_fem)
  end;

(* To get the male female split for Jr. Partners, use below *)
get_male_female ``Jr. Partner'' table;
```

3.2 Test Results

3.2.1 Test results for employee data

¿ ¿ ¿ val table = [("Nancy", "F", "Associate", 7), ("Peter", "M", "Associate", 5), ("Paul", "M", "Sr. Partner", 23), ("Robert", "M", "Jr. Partner", 11), ("Ralph", "M", "Associate", 4), ("Rhonda", "F", "Associate", 11), ("Sam", "M", "Associate", 5), ("Sara", "F", "Associate", 9), ("Alice", "F", "Sr. Partner", 27), ("Bob", "M", "Jr. Partner", 14), ("Eve", "F", "Associate", 9), ("John", "M", "Jr. Partner", 17), ("Gary", "M", "Associate", 6), ("Adam", "M", "Associate", 8), ("Carl", "M", "Associate", 2), ("Douglas", "M", "Jr. Partner", 14), ("Sally", "F", "Jr. Partner", 19), ("Joseph", "M", "Sr. Partner", 33), ("Mary", "F", "Jr. Partner", 18), ("Esther", "F", "Associate", 8), ("Frank", "M", "Associate", 7), ("Jane", "F", "Associate", 3), ("George", "M", "Jr. Partner", 15), ("Harry", "M", "Associate", 1), ("Robin", "F", "Associate", 8), ("David", "M", "Jr. Partner", 16), ("Jan", "F", "Associate", 6), ("Jack", "M", "Associate", 4), ("Ken", "M", "Associate", 9), ("Kathy", "F", "Associate", 1), ("Larry", "M", "Associate", 7), ("Mike", "M", "Jr. Partner", 12), ("Ted", "M", "Jr. Partner", 18), ("Terry", "M", "Sr. Partner", 30), ("Victor", "M", "Associate", 3), ("Vera", "F", "Sr. Partner", 28), ("Wilma", "F", "Associate", 7), ("Zack", "M", "Associate", 1), ("Debra", "F", "Associate", 12), ("Bonnie", "F", "Associate", 5), ("Dan", "M", "Jr. Partner", 17), ("Meredith", "M", "Jr. Partner", 17), ("Randal", "M", "Associate", 9), ("Jake", "M", "Jr. Partner", 12), ("Karen", "F", "Jr. Partner", 9)]; val table = [("Nancy", "F", "Associate", 7), ("Peter", "M", "Associate", 5), ("Paul", "M", "Sr. Partner", 23), ("Robert", "M", "Jr. Partner", 11), ("Ralph", "M", "Associate", 4), ("Rhonda", "F", "Associate", 11), ("Sam", "M", "Associate", 5), ("Sara", "F", "Associate", 9), ("Alice", "F", "Sr. Partner", 27), ("Bob", "M", "Jr. Partner", 14), ("Eve", "F", "Associate", 9), ("John", "M", "Jr. Partner", 17), ("Gary", "M", "Associate", 6), ("Adam", "M", "Associate", 8), ("Carl", "M", "Associate", 2), ("Douglas", "M", "Jr. Partner", 14), ("Sally", "F", "Jr. Partner", 19), ("Joseph", "M", "Sr. Partner", 33), ("Mary", "F", "Jr. Partner", 18), ("Esther", "F", "Associate", 8), ("Frank", "M", "Associate", 7), ("Jane", "F", "Associate", 3), ("George", "M", "Jr. Partner", 15), ("Harry", "M", "Associate", 1), ("Robin", "F", "Associate", 8), ("David", "M", "Jr. Partner", 16), ("Jan", "F", "Associate", 6), ("Jack", "M", "Associate", 4), ("Ken", "M", "Associate", 9), ("Kathy", "F", "Associate", 1), ("Larry", "M", "Associate", 7), ("Mike", "M", "Jr. Partner", 12), ("Ted", "M", "Jr. Partner", 18), ("Terry", "M", "Sr. Partner", 30), ("Victor", "M", "Associate", 3), ("Vera", "F", "Sr. Partner", 28), ("Wilma", "F", "Associate", 7), ("Zack", "M", "Associate", 1), ("Debra", "F", "Associate", 12), ("Bonnie", "F", "Associate",

5), ("Dan", "M", "Jr. Partner", 17), ("Meredith", "M", "Jr. Partner", 17), ("Randal", "M", "Associate", 9), ("Jake", "M", "Jr. Partner", 12), ("Karen", "F", "Jr. Partner", 9)]: (string * string * string * int) list

3.2.2 Test results for calch

1

```
> fun calc_h (status:string) [] = 0
  | calc_h (status) (head::table:(string*string*string*int)list) =
  let
    val (name, gender, j_status, exp) = head
    val match:bool = (status = j_status)
  in
    if match then (2080*exp)+(calc_h status table) else (calc_h status table)
  end;


calc_h "Jr. Partner" table;
# # # # # # # val calc_h = fn: string -> (string * string * string * int) list -> int
> > # # val it = 434720: int
```

There were a calculated number of 434,720 total accumulated hours of experience for all Jr. Partners. This is substantially more than the recommended amount.

3.2.3 Test results for male/female split

2

```
> (* Calculate the male/female split *)

fun get_male_female(status:string) [] = (0, 0)
  | get_male_female(status:string) (head::table) =
  let
    (* Use pattern matching to get elmenets of list entry *)
    val (name, gender, j_status, exp) = head
    val f_match:bool = ((status = j_status) andalso (gender = "F"))
    val m_match:bool = ((status = j_status) andalso (gender = "M"))
    val (p_male, p_fem) = get_male_female status table
  in
    ((if m_match then 1 else 0)+p_male,(if f_match then 1 else 0)+p_fem)
  end;

get_male_female "Jr. Partner" table;
# # # # # # # # # # # # val get_male_female = fn:
   string -> ('a * string * string * 'b) list -> int * int
> > # # val it = (11, 3): int * int
```

The split between the Jr. Partners is 11 males to 3 females, about 79 percent male. This is more than adequate for the gender ratio requirement. Hiring more Jr. Partners seems like a good long term plan due to the high experience and male gender ratio.

Appendix A

Source Code for Sample Exercise

The following code is from *quiz1.sml*

```
(
* Quiz 1 SML Code *)

(* Table is a (string * string * string * int) list for the records
val table =
[("Nancy", "F", "Associate", 7),
("Peter", "M", "Associate", 5),
("Paul", "M", "Sr. Partner", 23),
("Robert", "M", "Jr. Partner", 11),
("Ralph", "M", "Associate", 4),
("Rhonda", "F", "Associate", 11),
("Sam", "M", "Associate", 5),
("Sara", "F", "Associate", 9),
("Alice", "F", "Sr. Partner", 27),
("Bob", "M", "Jr. Partner", 14),
("Eve", "F", "Associate", 9),
("John", "M", "Jr. Partner", 17),
("Gary", "M", "Associate", 6),
("Adam", "M", "Associate", 8),
("Carl", "M", "Associate", 2),
("Douglas", "M", "Jr. Partner", 14),
("Sally", "F", "Jr. Partner", 19),
("Joseph", "M", "Sr. Partner", 33),
("Mary", "F", "Jr. Partner", 18),
("Esther", "F", "Associate", 8),
("Frank", "M", "Associate", 7),
("Jane", "F", "Associate", 3),
("George", "M", "Jr. Partner", 15),
("Harry", "M", "Associate", 1),
("Robin", "F", "Associate", 8),
("David", "M", "Jr. Partner", 16),
("Jan", "F", "Associate", 6),
("Jack", "M", "Associate", 4),
("Ken", "M", "Associate", 9),
("Kathy", "F", "Associate", 1),
("Larry", "M", "Associate", 7),
("Mike", "M", "Jr. Partner", 12),
("Ted", "M", "Jr. Partner", 18),
("Terry", "M", "Sr. Partner", 30),
("Victor", "M", "Associate", 3),
("Vera", "F", "Sr. Partner", 28),
("Wilma", "F", "Associate", 7),
```

```
("Zack", "M", "Associate", 1),
("Debra", "F", "Associate", 12),
("Bonnie", "F", "Associate", 5),
("Dan", "M", "Jr. Partner", 17),
("Meredith", "M", "Jr. Partner", 17),
("Randal", "M", "Associate", 9),
("Jake", "M", "Jr. Partner", 12),
("Karen", "F", "Jr. Partner", 9)];

(* Calculate the total hours of experience between all Jr. Partners *)
(* assume one year = 2080 hours of experience                       *)
fun calc_h (status:string) [] = 0
  | calc_h (status) (head::table:(string*string*string*int)list) =
  let
    val (name, gender, j_status, exp) = head
    val match:bool = (status = j_status)
  in
    if match then (2080*exp)+(calc_h status table) else (calc_h status table)
  end;

calc_h "Jr. Partner" table;

(* Calculate the male/female split *)

fun get_male_female(status:string) [] = (0, 0)
  | get_male_female(status:string) (head::table) =
  let
    (* Use pattern matching to get elmenets of list entry *)
    val (name, gender, j_status, exp) = head
    val f_match:bool = ((status = j_status) andalso (gender = "F"))
    val m_match:bool = ((status = j_status) andalso (gender = "M"))
    val (p_male, p_fem) = get_male_female status table
  in
    ((if m_match then 1 else 0)+p_male,(if f_match then 1 else 0)+p_fem)
  end;

get_male_female "Jr. Partner" table;
```

Appendix B

Addendum

B.1 Relevant Code

The following code is from *modquiz1.sml*

```
(
* Quiz 1 SML Modified Code *)

(* Table is a (string * string * string * int) list for the records
(* The table will remain the same *)
val table =
[("Nancy", "F", "Associate", 7),
("Peter", "M", "Associate", 5),
("Paul", "M", "Sr. Partner", 23),
("Robert", "M", "Jr. Partner", 11),
("Ralph", "M", "Associate", 4),
("Rhonda", "F", "Associate", 11),
("Sam", "M", "Associate", 5),
("Sara", "F", "Associate", 9),
("Alice", "F", "Sr. Partner", 27),
("Bob", "M", "Jr. Partner", 14),
("Eve", "F", "Associate", 9),
("John", "M", "Jr. Partner", 17),
("Gary", "M", "Associate", 6),
("Adam", "M", "Associate", 8),
("Carl", "M", "Associate", 2),
("Douglas", "M", "Jr. Partner", 14),
("Sally", "F", "Jr. Partner", 19),
("Joseph", "M", "Sr. Partner", 33),
("Mary", "F", "Jr. Partner", 18),
("Esther", "F", "Associate", 8),
("Frank", "M", "Associate", 7),
("Jane", "F", "Associate", 3),
("George", "M", "Jr. Partner", 15),
("Harry", "M", "Associate", 1),
("Robin", "F", "Associate", 8),
("David", "M", "Jr. Partner", 16),
("Jan", "F", "Associate", 6),
("Jack", "M", "Associate", 4),
("Ken", "M", "Associate", 9),
("Kathy", "F", "Associate", 1),
("Larry", "M", "Associate", 7),
("Mike", "M", "Jr. Partner", 12),
("Ted", "M", "Jr. Partner", 18),
("Terry", "M", "Sr. Partner", 30),
```

```
("Victor", "M", "Associate", 3),
("Vera", "F", "Sr. Partner", 28),
("Wilma", "F", "Associate", 7),
("Zack", "M", "Associate", 1),
("Debra", "F", "Associate", 12),
("Bonnie", "F", "Associate", 5),
("Dan", "M", "Jr. Partner", 17),
("Meredith", "M", "Jr. Partner", 17),
("Randal", "M", "Associate", 9),
("Jake", "M", "Jr. Partner", 12),
("Karen", "F", "Jr. Partner", 9)];

(* Calculate the total hours of experience between all Associates *)
(* assume one year = 2080 hours of experience                     *)
fun calc_h (status:string) [] = 0
  | calc_h (status) (head::table:(string*string*string*int)list) =
  let
    val (name, gender, j_status, exp) = head
    val match:bool = (status = j_status)
  in
    if match then (2080*exp)+(calc_h status table) else (calc_h status table)
  end;

(* Modifed line below for Associate *)
(* calc_h "Jr. Partner" table; *)
calc_h "Associate" table;

(* Calculate the male/female split *)

fun get_male_female(status:string) [] = (0, 0)
  | get_male_female(status:string) (head::table) =
  let
    (* Use pattern matching to get elmenets of list entry *)
    val (name, gender, j_status, exp) = head
    val f_match:bool = ((status = j_status) andalso (gender = "F"))
    val m_match:bool = ((status = j_status) andalso (gender = "M"))
    val (p_male, p_fem) = get_male_female status table
  in
    ((if m_match then 1 else 0)+p_male,(if f_match then 1 else 0)+p_fem)
  end;

(* Modifed line below for Associate *)
(* get_male_female "Jr. Partner" table; *)
get_male_female "Associate" table;
```

B.2 Test Cases

B.2.1 Test cases for modified calch

3

```
> fun calc_h (status:string) [] = 0
  | calc_h (status) (head::table:(string*string*string*int)list) =
  let
    val (name, gender, j_status, exp) = head
    val match:bool = (status = j_status)
  in
    if match then (2080*exp)+(calc_h status table) else (calc_h status table)
  end;

(* Modifed line below for Associate *)
(* calc_h "Jr. Partner" table; *)
calc_h "Associate" table;

val calc_h = fn: string -> (string * string * string * int) list -> int
> > val it = 326560: int > plus 0 0;
```

The Associates have a combined total of 326,560 hours of experience.

B.2.2 Test cases for modified male/female split

4

```
> (* Calculate the male/female split *)

fun get_male_female(status:string) [] = (0, 0)
  | get_male_female(status:string) (head::table) =
  let
    (* Use pattern matching to get elmenets of list entry *)
    val (name, gender, j_status, exp) = head
    val f_match:bool = ((status = j_status) andalso (gender = "F"))
    val m_match:bool = ((status = j_status) andalso (gender = "M"))
    val (p_male, p_fem) = get_male_female status table
  in
    ((if m_match then 1 else 0)+p_male,(if f_match then 1 else 0)+p_fem)
  end;

(* Modifed line below for Associate *)
(* get_male_female "Jr. Partner" table; *)
get_male_female "Associate" table;
# # # # # # # # # # # # val get_male_female = fn:string -> ('a * string * string * 'b) list -> int * int
> > val it = (14, 12): int * int
```

The Jr. Partners have a much higher ratio of males to females(11 to 3) than the Associates (14 to 12). In addition, the total hours of experience between the Jr. Partners is much higher than for the Associates. Hiring more Jr. Partners appears to be the sound option over more Associates to prevent risk of the male gender ratio dropping and stay above the recommendation.