CMPS 258 – PROGRAMMING LANGUAGES – SPRING 2021
WEEK 1 ASSIGNMENT

In this assignment, you will write simple SML functions that manipulate student grades in a class. Assume that a student's record is represented as a quadruple of type `int*string*real*real` which contains the students ID, name, midterm exam grade, and final exam grade, respectively.

(1) Implement a function `score` that takes a student record and computes the student's composite score, assigning 40% to the midterm and 60% to the final.

(2) Implement a function `scores` that takes a list of student records and returns a list of pairs of type `int*real` consisting of each student's ID and composite score, respectively.

(3) Implement a function `getByID` that takes a list of student records and an integer `i` and returns the record of the student whose ID is `i`. Assume the list contains the desired student (do not worry about the case where the desired student is not in the list).

(4) Implement a function `whoFailedMidterm` that takes a list of student records and returns a list of the names of the students who received less than 60 on the midterm exam.

(5) Implement a function `allPassedFinal` that takes a list of student records and returns true if all the students received a grade greater than or equal to 60 on the final exam, and false otherwise.

(6) Implement a function `countPassedCourse` that takes a list of student records and returns the number of students who received a composite score greater than or equal to 60.

(7) Implement a function `studentsInRange` that takes a list of student records and a pair of reals `range` and returns a list of the records of the students who received a composite score in between `#1 range` and `#2 range` (exclusive).

(8) Implement a function `countStudentsInRanges` that takes a list of student records and a list of pairs `ranges` and returns the number of students who received a composite score in at least one of the ranges in `ranges`. The ranges may overlap so be careful not to double-count students that fall in multiple ranges.

(9) Implement a function `studentWithHighestScore` that takes a list of student records and returns the record of the student with the highest composite score. Assume the class always has at least one student (do not check for the case when the list is empty).

(10) Implement a function `insertAssignmentGrade` that takes a list of student records and a list of assignment grades and returns a list of amended student records that include the assignment grade as a third field in the tuple (the returned list will have type `int*string*real*real*real`). Assume that the two lists are of equal size (do not check for the case where their size mismatches).

Evaluating a correct homework solution should generate the bindings below. However, keep in mind that generating these bindings does not guarantee that your solutions are correct. Make sure to test your functions before submitting.

```
val score = fn : int * string * real * real -> real
val scores = fn : (int * string * real * real) list -> (int * real) list
val getByID = fn
  : (int * string * real * real) list * int -> int * string * real * real
val whoFailedMidterm = fn : (int * string * real * real) list -> string list
val allPassedFinal = fn : (int * string * real * real) list -> bool
val countPassedCourse = fn : (int * string * real * real) list -> int
val studentsInRange = fn
  : (int * string * real * real) list * (real * real)
    -> (int * string * real * real) list
val countStudentsInRanges = fn
  : (int * string * real * real) list * (real * real) list -> int
val studentWithHighestScore = fn
  : (int * string * real * real) list -> int * string * real * real
val insertAssignmentGrade = fn
  : (int * string * real * real) list * real list
    -> (int * string * real * real * real) list
```

Assessment
Solutions should be:

- Correct
- In good style, including indentation and line breaks
- Written using features discussed in class

Do not use pattern-matching. It will be used in the next assignment.

Submission Instructions
Put all your solutions in one SML file and submit it via Moodle. The file should be named “<id>.sml” where <id> is your AUBnet ID (e.g., abc01.sml). Do not submit any other files or compressed folders.