The questions below require you to write ML functions. Some of the problems in this assignment require a mix of functional and non-functional programming, and specifically, the use of references, assignment statements and iteration. However, unless a question explicitly requires the use of such imperative programming features, your programs should be purely functional and should use recursion. Pattern matching should also be used whenever possible, unless it complicates the code significantly. In general, simple solutions are preferred and will receive the most marks. Feel free to use helper functions wherever appropriate. Unless otherwise specified, you may assume that the input to your functions is correct, so that no error checking is required. Finally, by properly raising exceptions, your functions in this assignment should not produce any warnings of the form `match nonexhaustive`. You may have to detect illegal inputs in order to avoid such warnings, and in such cases, you should print out an error message.

Unless specified otherwise, do not use any built-in functions that would require recursion if you defined them yourself. You may, of course, use any function you like if you define it yourself (in terms of allowed functions). The point here is that you should not scour the user manual or the web for functions that will solve most of a problem for you.

You should hand in four files: the source code of all your ML functions, a sample terminal session with the ML interpreter, the answers to pencil-and-paper problems, and a signed and completed cover sheet. The source code should be well commented, and the terminal session should be short and should demonstrate that your functions work correctly. These files should be submitted electronically using the submission web page.

**Note:** The marker has a limited amount of time for each assignment, so it is your responsibility to provide documentation and testing that allows him to quickly evaluate your work. As with all work in this course, 20% of the grade is for quality of presentation.

More questions will be added shortly
1. Basic Recursion and Pattern Matching (10 points total)

Using recursion, define an ML function \( \text{listSum}(A,X,Y) \) of type 
\[ \text{real*(real list)*(real list)} \rightarrow \text{(real list)} \]. If \( x_i \) is the \( i^{th} \) element of list \( X \), and \( y_i \) is the \( i^{th} \) element of list \( Y \), then \( A + x_i \times y_i \) is the \( i^{th} \) element of the output list. For example,

\[
\text{listSum}(7.3, [3.1, 4.2, 5.7], [2.7, 4.1, 1.5])
\]

\[
=> [7.3 + 3.1 \times 2.7, 7.3 + 4.2 \times 4.1, 7.3 + 5.7 \times 1.5]
\]

\[
=> [15.67, 24.52, 15.85]
\]

Do not use any \text{map} functions in your solution. Define the function in two ways:
(a) without pattern matching (5 points), and (b) with pattern matching (5 points). These two versions of the function should be called \text{listSum1} and \text{listSum2}, respectively.

2. Record Types and Exceptions (56 points total)

(a) (3 points)
Using the ML \text{type} command, define \text{student} to be a named type for student recorda, where each record has three fields: \text{id}, \text{name} and \text{gpa}, of type \text{int}, \text{string} and \text{real}, respectively. For example, \{\text{id}=1234, \text{name}='Spock', \text{gpa}=97.1\} is a record of type \text{student}. This record means that the student with \text{id}=1234 has name Spock and a gpa of 97.1.

(b) (3 points)
Likewise, define \text{taken} to be a record with three fields: \text{course}, \text{student} and \text{grade}, of type \text{string}, \text{int} and \text{real}, respectively. For example, \{\text{course}='csc324', \text{student}=1234, \text{grade}=87.2\} is a record of type \text{taken}. This record means that the student with \text{id}=1234 has taken course \text{csc324} and received a grade of 87.2.

(c) (5 points)
Define an ML function \text{updateGPA}(G,S) of type \text{real*student -> student} that changes the \text{gpa} of student \text{S} to \text{G}. i.e., the function returns a copy of \text{S} with the \text{gpa} field changed. Raise an exception if \text{G} is negative.

In the questions below, \text{Slist} can be an arbitrary list of student records, and \text{Tlist} can be an arbitrary list of \text{taken} records.

(d) (10 points)
Define an ML function \text{updateGrade}(T,Tlist) of type \text{taken*(taken list) -> (taken list)}. This function updates the grade a student received in a given course. Specifically, if \( T = \{\text{course}=C, \text{name}=N, \text{grade}=G\} \), then the function searches \text{Tlist} for a record with \text{course}=C and \text{name}=N. It then changes the grade in this record to \text{G}. That is, the function returns a copy of \text{Tlist} in which the grade has been changed in the appropriate record. If \text{Tlist} does not contain such a record, then an exception should be raised.
(e) (20 points)
Define an ML function `names(C,Slist,Tlist)` of type `string*(student list)*(taken list) -> (string list)`. This function returns the names of all students who have taken course `C`, according to the data in `Slist` and `Tlist`. Raise an exception if a student who has taken the course is not listed in `Slist`.

(f) (15 points)
Define an ML function `computeGPA(I,Tlist)` of type `int*(taken list) -> real`. This function computes the gpa of the student with `id=I` from the courses he has taken, as given in `Tlist`. If the student has not taken any courses, then an exception is raised.

3. Exception Handling (20 points)
This question builds on the previous one. Define an ML function `updateAllGPAs(Slist,Tlist)` of type `(student list)*(taken list) -> (student list)*(student list)`. This function uses `computeGPA` from the previous question to compute the gpa of each student in `Slist`. If `computeGPA` raises an exception, then `updateAllGPAs` catches and handles it by putting the student on a list of exceptional students. Otherwise, `updateAllGPAs` uses `updateGPA` to produce an updated student record containing the computed gpa, and puts this record onto a list of updated student records. Finally, `updateAllGPAs` returns both the list of updated student records and the list of exceptional records, in that order. Every student in the input list should appear in one of the two output lists.

More questions will be added shortly
Cover sheet for Assignment 2

Complete this page and submit it with your assignment.

Name: ____________________________
(Underline your last name)

Student number: ______________________

I declare that this assignment is solely my own work, and is in accordance with the University of Toronto Code of Behaviour on Academic Matters.

Signature: ____________________________