CSCI3180 Assignment 4

Introduction

Declarative programming is a programming paradigm that emphasises the expression of "what" the problem is over "how" to solve the problem. You will gain experience of declarative programming with ML (Functional Programming) in this assignment. The assignment consists of two parts:

- Exercise on ML.
- Bullfighting Poker Game.

You will submit one source code file for each part at the end. No written report is needed this time.

You are supposed to test your work on the machine **sparc1.cse.cuhk.edu.hk** using Standard ML of New Jersey, Version 110.99.4, which can be invoked using the commands `sml`.

Read the pinned Assignment 4 FAQ posts on Piazza for some common clarifications. You should check that a day before the deadline to make sure you don't miss any clarifications, even if you have already submitted your work then.

If you need further clarification of the requirements, please feel free to post on the Piazza with the assignment4 tag. However, to avoid cluttering the forum with repeated/trivial questions, please do read all the given code, webpage description, sample output, and latest FAQ carefully before posting your questions. Also, please be reminded that we won't debug for any student's assignment for the sake of fairness and that you should not share/post your solution code anywhere.

About academic integrity

We value academic integrity very highly.

- Do NOT try your "luck" - we use sophisticated plagiarism detection software to find cheaters.
- The penalty (for BOTH the copier and the copiee) may not be just getting a zero in your assignment. There may be additional disciplinary actions from the university, upto and including expulsion.
- Do NOT copy or look at the assignment code/work of another student/person.
- Do NOT share your assignment code/work to anyone and do NOT post it anywhere online.
- While we encourage discussion among students, you should write the code and finish the work on your own. In the discussion, you should NOT go into the details such that everyone will end up with the same code/work.
- You are responsible to double-check your own submission and do your best to avoid "accidents".

Read https://www.cuhk.edu.hk/policy/academichonesty/ for details of the CUHK policy on this matter.

Part one: Exercise on ML

Implement the predicates or queries of the following problems in a ML program file named **a4.sml**. The corresponding question number of each problem should be clearly indicated using comments.

1. Efficient list reversing. Note that you can get full marks for Q1 as long as your implementation of Q1 (b) is correct.
   - (a) Define function `reverse(lst)` which returns the reversed version of list `lst`. Note that you are **NOT** allowed to use built-in function `rev`. Example usage:

     ```
     fun reverse lst : int list = ;
     (* Your implementation here. Feel free to modify the definition here for pattern matching if needed. *)

     - reverse [];
     val it = [] : int list;

     - reverse [3, 1, 8, 0];
     val it = [0, 8, 1, 3] : int list
     ```

     Hint: you may find `@` operator helpful in this task.
   - (b) Re-implement the above program, but you are **NOT** allowed to use function `rev` and operator `@`.
2. Partial derivative of bivariate polynomials: recall that we have briefed how to express bivariate polynomials using ML on Page 32, Tutorial 9.

   ```
   (* constructers *)
   datatype term = Term of int * int * int;
   datatype variable = Variable of string;
   datatype poly = Poly of variable * variable * term list;

   (* For example: 3x^3y + x^2 - 4y^2 + 2xy + 1 *)
   (* Poly(Variable "x", Variable "y", [Term(3, 1, 3), Term(2, 0, 1), Term(0, 2, ~4), Term(1, 1, 2), Term(0, 0, 1)]) *)
   (* Note that we assume x and y are two variables, and the 1st and 2nd elements of term are the exponentials of x and y, respec

   (* Several functions you may find helpful for computations over polynomials *)
   fun expon_x (Term(e, _, _)) = e;
   fun expon_y (Term(_, e, _)) = e;
   fun coeff (Term(_, _, c)) = c;
   ```

   In this question, we focus on the partial derivative of bivariate polynomials with respect to the two variables x and y. A brief introduction on the basic rules for the derivative of polynomials can be found here. In this task, you are required to implement a function `diff_poly(p:poly, v:variable)` which calculates the partial derivative of a bivariate polynomial `p` with respect to the variable `v`. Example usage:

   ```
   exception VariableMismatch;

   fun diff_terms (l : term list, v : variable) : term list = ;
   (* Your implementation here *)

   fun diff_poly (Poly(xx, yy, l), v) : poly =
     if (xx = v) orelse (yy = v) then
       Poly (xx, yy, diff_terms (l, v))
     else
       raise VariableMismatch
   ;

   (*
   * We assume the all polynomials are bivariate with variables x and y.
   * And all the inputs strictly follow the data structures defined above.
   * You are not required to raise any exceptions or do any type checking in the implementation.
   *)
   val x = Variable "x";
   val y = Variable "y";

   (* Test cases *)
   val p = Poly(x, y, [Term(3, 1, 3), Term(2, 0, 1), Term(0, 2, ~4), Term(1, 1, 2), Term(0, 0, 1)]);

   - diff_poly (p, x);
   val it = Poly (Variable "x", Variable "y", [Term (2, 1, 9), Term (1, 0, 2), Term (0, 1, 2)]) : poly;

   - diff_poly (p, y);
   val it = Poly (Variable "x", Variable "y", [Term (3, 0, 3), Term (0, 1, ~8), Term (1, 0, 2)]) : poly
   ```

Part two: Bullfighting Poker Game

Implement the required functions of the following problems in a ML program file named **a4.sml**. The corresponding question number of each problem should be clearly indicated using comments.

Background

We design a two-player poker game named buillfighting for the practice of functional programming. We consider all the cards except for joker cards in this game, and only the ranks of cards are considered. Each rank has its corresponding point. All the ranks with the corresponding points are listed as follows:

```
Ace=1, King=10, Queen=10, Jack=10, 10, 9, 8, 7, 6, 5, 4, 3, 2
```

For simplity, we use number 1 to 10 to represent all the ranks. In the bullfighting game, each player will get five cards, respectively. And the outcome of the game is determined by the following rules:

- If the total points of any combination formed by three cards out of the five held by a player are divisible by 10, we call it a **Bull**. The player's point is then determined by the remainder of the sum of the points of the remaining two cards divided by 10. If the remainder is 0, *i.e.*, the sum is divisible by 10, the player's point is 10.
- Otherwise, the player's point is determined as the point of the card with the highest point.

Note that Bulls always beat non-Bulls. We compare the points of players only when both players are both Bull or both not Bull, and the player with higher points wins. If both the players have the same points, this is a tie. Here are some examples to illustrate the rules of the game:

```
# Case 1
Player 1: [10, 2, 6, 2, 4], is a Bull (10 + 6 + 4 = 20, which is divisible by 10) with points 4 (2 + 2).
Player 2: [3, 7, 4, 6, 5], is not a Bull (no combination of 3 cards is divisible by 10) with points 7.
Player 1 wins.

# Case 2
Player 1: [10, 10, 10, 3, 5], is a Bull (10 + 10 + 10 = 30) with points 8 (3 + 5).
Player 2: [3, 10, 4, 9, 3], is a Bull (3 + 4 + 3 = 10) with points 9 (10 + 9).
Player 2 wins.

# Case 3
Player 1: [10, 10, 5, 2, 4], is not a Bull with points 10.
Player 2: [2, 1, 8, 5, 6], is not a Bull with points 8.
Player 1 wins.

# Case 4
Player 1: [10, 6, 2, 2, 8], is a Bull (10 + 2 + 8 = 20) with points 8 (6 + 2).
Player 2: [7, 3, 1, 6, 1], is a Bull (3 + 1 + 6 = 10) with points 8 (7 + 1).
This is a tie.
```

Tasks

A set of functions should be implemented to help you finish the main program step by step.

1. Implement a function `check_bull(lst)` that takes a list of five cards for a player as input. The function returns a bool indicating whether the input list is a Bull. Hints: check all combinations and determine if there is a combination that meets the condition. For example:

   ```
   check_bull [10, 2, 6, 2, 4];      (* return: true *)
   check_bull [3, 7, 4, 6, 5];       (* return: false *)
   ```

2. Implement a function `get_point_bull(lst)` that takes a list of five cards for a player as input. The function returns an int indicating the points of the player when the list is a Bull. We assume the input to this function is a Bull, so you don't need to check whether the input is Bull in this function. For example:

   ```
   get_point_bull [10, 2, 6, 2, 4];     (* return: 4 *)
   get_point_bull [7, 3, 10, 2, 8];     (* return: 10 *)
   ```

3. Implement a function `get_point_non_bull(lst)` that takes a list of five cards for a player as input. The function returns an int indicating the points of the player when the list is not a Bull. We assume the input to this function is not a Bull, so you don't need to check whether the input is Bull in this function. For example:

   ```
   get_point_non_bull [3, 7, 4, 6, 5];      (* return: 7 *)
   ```

4. Implement a function `compare_result(lst1, lst2)` that takes two players' lists as input. The function returns a string selected from three candidates, *i.e.*, "Player 1 wins", "Player 2 wins" and "This is a tie". For example:

   ```
   val lst1 = [10, 2, 6, 2, 4];
   val lst2 = [3, 7, 4, 6, 5];
   compare_result (lst1, lst2);     (* return: "Player 1 wins" *)
   ```

Submission

Submission deadline: 23:59:00, Apr 23 (Tuesday)

Please submit your work as **a4.sml** to Blackboard "Assignment 4". Do NOT zip/compress/archive it.

Make sure you have added and completed the following declaration at the top of the file:

```
(*
* CSCI3180 Principles of Programming Languages
*
* --- Declaration ---
* I declare that the assignment here submitted is original except for source
* materials explicitly acknowledged. I also acknowledge that I am aware of
* University policy and regulations on honesty in academic work, and of the
* disciplinary guidelines and procedures applicable to breaches of such policy
* and regulations, as contained in the website
* http://www.cuhk.edu.hk/policy/academichonesty/
*
* Name:
* Student ID:
* Email Address:
*
* Source material acknowledgements (if any):
*
* Students whom I have discussed with (if any):
*
*)
```

The usual assignment submission policy applies. The official testing environment is Standard ML of New Jersey, Version 110.99.4. Refer to the tutorials for how to use SSH to access the sparc1 Linux server which has that installed. On the server, you can execute `sml` to launch it and load your code with `use "a4.sml";` command.