COMP 3031 Assignment 1
SML programming
Fall 2021

Due: 5PM on 7 October 2021 (Thursday)

Instructions

- There are **five** questions in this assignment. Each question counts for two points. The total number of points is 10.
- This is an individual assignment. You can discuss with others and search online resources, but your submission should be your own code.
- Write your functions exactly the same as defined in the problem description (name, type, and functionality). In addition, you can write any helper functions as needed and call any built-in SML functions available in the lab machine.
- Put your entire solution in a single text file called "*ass1.ml*". In this file, put down your **name, ITSC account, and student ID** as a ***comment*** (surrounded by "(*" and "*)") on the first line.
- Submit your file through the Canvas before the deadline.
- Your submission will be tested under the SML interpreter on a lab machine by issuing the following command:

- *use "ass1.ml";*

- Assume all test cases are valid as specified in the function description.
- No late submissions will be accepted under usual circumstances.

NOTE: We will perform code similarity checks. In case a submission has code similarity issues, we may request clarification and deduct partial marks or full marks on a case-by-case basis.

We define the following data types to be used throughout Assignment 1:

```
datatype flight = F of int * int;
datatype flights = Fs of flight list;
```

The data type `flight` is constructed on a two-element tuple consisting of integers `x` and `y` representing a direct flight from a city x to another city y. Assume `x` and `y` in the tuple are of different values. For example, the tuple `(0,1)` represents a direct flight from city `0` to city `1`. The data type `flights` is constructed on a list of elements of the `flight` type. Assume all tuples in the `flight` list are unique.

Question 1.

Write a function `reachable` that checks whether we can fly from city `a` to city `b` via a direct flight or connecting flights in `flights`. If it is reachable from city `a` to city `b`, the function will return true; otherwise, it will return false.

```
val reachable = fn : flights * (int * int) -> bool
```

Examples:

```
- reachable (Fs [], (0, 1));
val it = false : bool


- reachable (Fs [F(0,1), F(1,0)], (0, 1));
val it = true : bool


- reachable (Fs [F(0,1), F(1,2), F(2,0)], (0, 2));
val it = true : bool
```

Question 2.

Write a function `popular_cities` that returns a list of ids of cities that are the destinations of the greatest number of direct flights. If more than one city have the greatest number of incoming direct flights, the function will return the list of ids of all these cities. The city ids in the output list are unique, but the order they appear in the output list is unimportant.

```
val popular_cities = fn : flights -> int list
```

Examples:

```
- popular_cities (Fs []);
val it = [] : int list
```

```
- popular_cities (Fs [F(0,1), F(1,0)]);
val it = [0,1] : int list


- popular_cities (Fs [F(0,1), F(0,2), F(1,0), F(2,1)]);
val it = [1] : int list
```

Question 3.

Write a function `count_paths` that returns the number of paths (containing one or more than one flight) in `flights` from city `src` to city `dst` in the input tuple `(src, dst)`. All cities in each counted path must be unique, i.e., no city appears more than once in the path.

```
val count_paths = fn : flights * (int * int) -> int
```

Examples:

```
- count_paths (Fs [], (0, 1));
val it = 0 : int


- count_paths (Fs [F(0,1), F(1,0)], (0, 1));
val it = 1 : int


- count_paths (Fs [F(0,1), F(0,2), F(1,2), F(1,3),
F(2,0), F(3,2)], (0, 2));
val it = 3 : int
```

Question 4.

Write a function `shortest_paths` that returns a list containing all the shortest paths from city `src` to city `dst` in the given `flights`. If more than one path is the shortest, the function will return all these paths. If there is no path from city `src` to city `dst`, the function will return an empty list. The paths in the output list are unique, but the order these paths appear in the list is unimportant.

```
val shortest_paths = fn : flights * (int * int) -> int
list list
```

Examples:

```
- shortest_paths (Fs [], (0, 1));
val it = [] : int list list
```

```
- shortest_paths (Fs [F(0,1), F(1,0)], (0, 1));
val it = [[0,1]] : int list list


- shortest_paths (Fs [F(0,1), F(0,3), F(1,2),F(1,3),
F(2,0), F(3,2)], (0, 2));
val it = [[0, 1, 2],[0, 3, 2]] : int list list
```

Question 5.

Write a function `search_cities` that returns a list of city pairs `(a,b)` if it is reachable from `a` to `b` within the given path length `L`. All city pairs in the output list are unique, but the order of the city pairs in the output list is unimportant.

```
val search_cities = fn : flights * int -> (int * int)
list
```

Examples:

```
- search_cities(Fs [], 1);
val it = [] : (int * int) list


- search_cities(Fs [F(0,1), F(1,0)], 1);
val it = [(0,1), (1,0)] : (int * int) list


- search_cities (Fs [F(0,2), F(1,0), F(2,1)], 2);
val it = [(0,2),(0,1),(2,1),(2,0),(1,0),(1,2)] : (int *
int) list
```