CS 496: Assignment 1
Due: February 9th, 11:55pm

1 Assignment Policies

Collaboration Policy. Homework will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems or programming. Use of the Internet is allowed, but should not include searching for existing solutions.

Under absolutely no circumstances code can be exchanged between students. Excerpts of code presented in class can be used.

Assignments from previous offerings of the course must not be re-used. Violations will be penalized appropriately.

2 Assignment

This assignment is about coding with lists and tuples. The setting is that of a simple game called *Connect the Dots*. A coded picture is a list of coordinates such that adjacent coordinates are aligned. Coordinates must be positive integers. Two coordinates are said to be **aligned** if they are either on the same x-axis, the same y-axis, or the difference between their x and y coordinates is the same (i.e. "diagonal").

A coded picture induces a picture, namely the one obtained by connecting the coordinates in the list, in the order they are presented. For example, given the coded picture

```
[(0,0);(0,4);(2,2);(0,0)]
```

it induces a triangle.

3 Exercises

We will use the following user defined datatypes for representing coordinates and coded pictures.

```
1 type coord = int*int
2 type coded_picture = coord list
```

Below are two examples of coded pictures, the first one determines a box (once the coordinates are connected) and the second a triangle.

```
let cp1 = [(0,0);(2,0);(2,2);(0,2);(0,0)]
let cp2 = [(0,0);(4,0);(4,4);(0,0)]
```

3.1 Your Task

Implement the following functions. You should work on the file `cp_stub.ml`. First rename it to `cp.ml` and then start completing the exercises given below.

1. `stretch : coded_picture -> int -> coded-picture`

   The expression `stretch cp factor` enlarges the coded picture by a factor of `factor` in both axis. For example,

   ```
   utop # stretch cp1 2;;
   - : coded_pic = [(0, 0); (4, 0); (4, 4); (0, 4); (0, 0)]
   utop # stretch cp2 7;;
   - : coded_pic = [(0, 0); (28, 0); (28, 28); (0, 0)]
   ```

   Provide two versions:

   (a) `stretch` using explicit recursion; and

   (b) `stretch_m` using map.

2. `segment : coord -> coord -> coord list`

   Given aligned coordinates `c1` and `c2`, it returns all the intermediate coordinates in the line segment from `c1` to `c2`, excluding `c1` but including `c2`. For example,

   ```
   # segment (1,1) (4,4);;
   - : (int * int) list = [(2, 2); (3, 3); (4, 4)]
   utop # segment (2,2) (5,2);;
   - : (int * int) list = [(3, 2); (4, 2); (5, 2)]
   ```

   Hint: The predefined function `compare: 'a -> 'a -> int` allows you to compare values as a prefix operation. In particular, `compare x 0` behaves like sign.

3. `coverage: coded_pic -> coord list`

   ```
   # coverage cp1;;
   - : coord list =
   [(0, 0); (1, 0); (2, 0); (2, 1); (2, 2); (1, 2); (0, 2); (0, 1); (0, 0)]
   # coverage cp2;;
   - : coord list =
   [(0, 0); (1, 0); (2, 0); (3, 0); (4, 0); (4, 1); (4, 2); (4, 3); (4, 4);
    (3, 3); (2, 2); (1, 1); (0, 0)]
   ```

   Provide two versions:

   (a) `coverage` using explicit recursion; and

(b) `coverage_f` using fold.

4. `equivalent_pics : coded_pic -> coded_pic -> bool`

   Determines whether two coded pictures are equivalent. Two coded pictures are said to be **equivalent** if the coordinates they cover are identical, up to possible duplicates. For example,

```
utop # equivalent_pics cp1  [(0,0);(1,0);(2,0);(2,2);(0,2);(0,0)];;
- : bool = true
utop # equivalent_pics cp1  [(2,2);(0,2);(0,0);(2,0);(2,2)];;
- : bool = true
```

5. `height : coded_pic -> int` and `width : coded_pic -> int`

   Determines the height (maximum difference between y coordinates) and width of a picture (maximum difference between x-coordinates). For example,

```
# height cp1;;
- : int = 2
utop # height cp2;;
- : int = 4
utop # width cp2;;
- : int = 4
utop # width cp1;;
- : int = 2
```

6. `tile : coded_pic -> int*int -> coded_pic list list`

   Given a coded picture and a pair of two numbers `n` and `m`, it returns a list of list of pictures. The result is obtained by constructing a first row of `n` copies, side by side of the original pictures, and then `m` rows below that. For example,

```
utop # tile (2,3) cp2;;
- : coded_pic list list =
[[[(0, 0); (4, 0); (4, 4); (0, 0)]; [(4, 0); (8, 0); (8, 4); (4, 0)]];
 [[(0, 4); (4, 4); (4, 8); (0, 4)]; [(4, 4); (8, 4); (8, 8); (4, 4)]];
 [[(0, 8); (4, 8); (4, 12); (0, 8)]; [(4, 8); (8, 8); (8, 12); (4, 8)]]]
utop # tile (2,1) cp2;;
- : coded_pic list list =
[[[(0, 0); (4, 0); (4, 4); (0, 0)]; [(4, 0); (8, 0); (8, 4); (4, 0)]]]
utop # tile (2,0) cp2;;
- : coded_pic list list = []
```

7. `tri_aligned: coord -> coord -> coord -> bool`

   Determines whether three coordinates are aligned, horizontally, vertically or in diagonal. For example,

```
utop # tri_aligned (0,0) (4,4) (9,9);;
- : bool = true
utop # tri_aligned (0,0) (0,4) (9,9);;
- : bool = false
utop # tri_aligned (0,0) (0,7) (0,9);;
- : bool = true
utop # tri_aligned (7,7) (2,2) (0,0);;
- : bool = true
```

8. `compress: coded_pic -> coded_pic`

   Compresses a coded picture by eliminating coordinates `c2` such that `c2` occurs between two other coordinates `c1` and `c3` and `c1`, `c2`, `c3` are aligned. For example,

```
# compress [(0,0);(1,0);(2,0);(2,2);(0,2);(0,0)];;
- : coded_pic = [(0, 0); (2, 0); (2, 2); (0, 2); (0, 0)]
```

4 Submission instructions

Submit a single file named `cp.ml` through Canvas. No report is required. Your grade will be determined as follows:

- You will get 0 points if your code does not compile.
- Partial credit may be given for style, comments and readability.