課題 2

1. 整数のリストの要素の最大値を返す関数 max を

```
let rec max a = match a with
  | [] -> min_int
  | h::[] -> h
  | h::t  -> if h > max t then h else max t;;
```

と書くと，最後の行で再帰呼出しを2回行うことがあるので大変効率が悪い.

   a. リスト長が n のとき、最大値を求めるために max は最小何回，最大何回呼び出されるか?
   b. O(n) の手間で最大値が求まるように max を改良する.

課題 2（つづき）

2. `List.fold_right` を使って
   - リストの要素の最大値を求める関数
   - リストとリストとの連結関数 (append)
   - マップ関数 (p.64)

   を (再帰呼出しを直接使わずに) 定義する.

ヒント: `a::b` の `::` はコンストラクタであって関数ではないので，`(::)` は高階関数の引数として渡すことはできない.

課題 2（つづき）

3. `List.fold_left` を使って
   - リストの長さを求める関数
   - リストの反転関数 (5/15演習)

   を (再帰呼出しを直接使わずに) 定義する.

課題 2（つづき）

4. `List.combine` と `List.fold_right`（または `List.fold_left`）を使って, 長さの等しい2本の整数リストの内積を計算する関数 `inner_product` を定義する.

課題 2（つづき）

5. ２本のリスト，たとえば `[1;2;3]` および `["a";"b"]` を受け取って，両者の「直積」，つまり

   `[(1,"a");(1,"b");(2,"a");(2,"b");(3,"a");(3,"b")]`

   を返す関数 `product` を，`List.map` 関数を利用して定義する.

6. ２本のリスト $[a_1; \ldots; a_m]$ および $[b_1; \ldots; b_n]$ と２引数関数 f を受け取って，集合

   $$\{ f(a_i, b_j) \mid 1 \leq i \leq m, 1 \leq j \leq n \}$$

   の各要素からなるリストを返す関数 `map_product` を定義する

課題 2（つづき）

7. 文字列のリストのリストを受け取って，各文字列が何番目のリストに出現するかを表す索引を作成する関数 index を定義する．たとえば

```
[["red";"green";"blue"];
 ["light-blue";"blue";"dark-blue"];
 ["pink";"orange";"red"]]
```

を受け取ったら index は以下のような結果を返せばよい．

```
[("blue",1); ("blue",2); ("dark-blue",2); ("green",1);
("light-blue",2); ("orange",3); ("pink",3); ("red",1);
("red",3)]
```

課題 2（つづき）

8. 前問の index 関数を改良して，リスト

```
[["red";"green";"blue"];
 ["light-blue";"blue"; "dark-blue"];
 ["pink";"orange";"red"]],
```

が与えられたら以下のようなリストが返るようにする.

```
[("blue", [1;2]); ("dark-blue", [2]); ("green", [1]);
 ("light-blue", [2]); ("orange", [3]); ("pink", [3]);
 ("red", [1;3])]
```

ヒント: いくつかの方法が考えられる.

課題 2

問題 1. ～ 8. について,

(a) 書いたプログラム（複数通りの解答も歓迎）

(b) 実行結果

- OCaml 処理系の出力の切り貼り（画面キャプチャではなく出力テキストの切り貼り）を，個々の問題に対して複数個添付する

(c) 感想

を CourseN@vi から提出してください.