2020 봄학기 프로그래밍언어
프로그래밍 과제 #4 (100점)

지수환 93suhwan@gmail.com
임현승 hsim@kangwon.ac.kr

제출 마감: 6월 1일 월요일 11:59pm

1 개요

- 숙제를 진행하기에 앞서 본 문서를 꼼꼼히 읽어보시길 바랍니다.
- 숙제에 대해서 친구들과 토론하는 것은 괜찮지만 숙제는 반드시 혼자서 작성하시기 바랍니다. **특히 코딩하는 중에 토론을 병행하거나 숙제를 완성한 이후에 다른 학생들에게 도움을 줄 경우 자칫하면 서로 매우 유사한 답안이 도출되어 표절로 판단될 수 있으니 주의하시기 바랍니다. 이 경우 설사 친구에게 소스코드를 보여주지 않았더라도 표절로 판단합니다.** 또한 다른 친구의 소스코드를 직접 보고 고쳐주는 경우도 부정행위에 해당합니다. 숙제가 표절로 판단될 경우 정보를 제공한 사람(copyee)과 정보를 도용한 사람(copier) 모두 0점 처리합니다. 여러분이 제출한 숙제는 프로그램 표절 검사기(clone-checker)를 이용해서 표절 여부를 검사합니다.
- 숙제가 어렵거나 이해가 안가는 부분이 있다면 가급적 과목 웹페이지 질의응답 게시판을 활용하세요. TA나 제가 최대한 친절히 도와드리겠습니다. 단, 정답은 가르쳐 드리지 않습니다.

숙제를 진행하기 위해 먼저 과목 웹페이지(`http://eruri.kangwon.ac.kr`) 과제 게시판에서 `hw4.zip` 파일을 다운받아 압축을 풉니다. 다운받은 파일 중에서 반드시 `hw4.ml` 파일만 수정하세요. `hw4.ml` 파일은 다음과 같습니다:

```
exception NotImplemented

let rec list_add _ _ = raise NotImplemented
let rec insert _ _ = raise NotImplemented
let rec insort _ = raise NotImplemented
let rec ltake _ _ = raise NotImplemented
let rec lall _ _ = raise NotImplemented
let rec lmap _ _ = raise NotImplemented
let rec lfilter _ _ = raise NotImplemented
let rec ltabulate _ _ = raise NotImplemented
let rec lrev _ = raise NotImplemented
let rec lconcat _ = raise NotImplemented
let rec lfoldl _ _ _ = raise NotImplemented
let rec lzip _ _ = raise NotImplemented
let rec split _ = raise NotImplemented
let rec cartprod _ _ = raise NotImplemented
```

본 숙제에서는 OCaml 표준 `list` 타입에 대해 여러가지 함수를 직접 구현합니다. 각 함수를 작성하기 위해 와일드카드 패턴 `_`를 적절한 변수 이름으로 변경하고, 등호의 오른쪽에 있는 `NotImplemented`를 삭제하고 문제가 요구하는 올바른 함수 정의를 채워 넣습니다.

2 문제

본 숙제에서는 OCaml `List` 라이브러리에서 제공하는 함수를 사용하면 안 됩니다. 반드시 모두 직접 구현해야 합니다. **List 라이브러리 함수를 이용할 경우 본 숙제는 0점입니다.**

2.1 `list_add` for adding each pair of integers from two lists [7점]

(타입) `list_add: int list -> int list -> int list`

(설명) `list_add` [a; b; c; ...] [x; y; z; ...] returns [$a+x$; $b+y$; $c+z$; ...].
If one list is longer than the other, the remaining list of elements is appended to the result.

(실행 예) `list_add [1; 2] [3; 4; 5]` returns `[4; 6; 5]`.

2.2 `insert` for inserting an element into a sorted list [7점]

(타입) `insert: int -> int list -> int list`

(설명) `insert` m l inserts m into a sorted list l. The resultant list is also sorted.

(가정) The list l is sorted in ascending order.

(실행 예) `insert 3 [1; 2; 4; 5]` returns `[1; 2; 3; 4; 5]`.

2.3 `insort` for insertion sort [7점]

(타입) `insort: int list -> int list`

(설명) `insort` l returns a sorted list of elements in l.

(실행 예) `insort [3; 7; 5; 1; 2]` returns `[1; 2; 3; 5; 7]`.

(Hint) Use `insert` above.

2.4 `ltake` for taking the list of the first n elements of l [7점]

(타입) `ltake: 'a list -> int -> 'a list`

(설명) `ltake` l n returns the list of the first n elements of l.
If `n` is larger than the length of `l`, then return `l`.

(실행 예) `ltake [3; 7; 5; 1; 2] 3` returns `[3; 7; 5]`.
`ltake [3; 7; 5; 1; 2] 7` returns `[3; 7; 5; 1; 2]`.
`ltake ["h"; "y"; "e"; "o"; "n"; "s"; "e"; "u"; "n"; "g"] 5` returns
`["h"; "y"; "e"; "o"; "n"]`.

(가정) $n \geq 0$

2.5 `lall` for examining a list [7점]

(타입) `lall : ('a -> bool) -> 'a list -> bool`

(설명) `lall` f l returns `true` if for every element x of l, f x evaluates to `true`; otherwise it returns `false`. In other words, `lall` f l tests if all elements in l satisfy the predicate f.

(실행 예) `lall (fun x -> x > 0) []` evaluates to `true`.
`lall (fun x -> x > 0) [1; 2; 3]` evaluates to `true`.
`lall (fun x -> x > 0) [1; -2; 3]` evaluates to `false`.

2.6 `lmap` for converting a list into another list [7점]

(타입) `lmap : ('a -> 'b) -> 'a list -> 'b list`

(설명) `lmap` f l applies f to each element of l from left to right, returning the list of results.

(실행 예) `lmap (fun x -> x + 1) [1; 2; 3]` returns `[2; 3; 4]`.

2.7 `lfilter` for filtering a list [7점]

(타입) `lfilter : ('a -> bool) -> 'a list -> 'a list`

(설명) `lfilter` p l returns every element of l that satisfies the predicate p.

(실행 예) `lfilter (fun x -> x > 2) [0; 1; 2; 3; 4; 5]` returns `[3; 4; 5]`.

2.8 `ltabulate` [7점]

(타입) `ltabulate : int -> (int -> 'a) -> 'a list`

(설명) `ltabulate` n f applies f to each element of a list `[0; 1; ...; n-1]`.

(실행 예) `ltabulate 4 (fun x -> x * x)` returns `[0; 1; 4; 9]`.

(가정) $n \geq 0$

2.9 `lrev` for reversing a list [7점]

(타입) `lrev: 'a list -> 'a list`

(설명) `lrev` l reverses l.

(실행 예) `lrev [1; 2; 3; 4]` returns `[4; 3; 2; 1]`.

2.10 `lconcat` for concatenating a list of lists [7점]

(타입) `lconcat : 'a list list -> 'a list`

(설명) `lconcat` l concatenates all elements of l.

(실행 예) `lconcat [[1; 2; 3]; [6; 5; 4]; [9]]` returns `[1; 2; 3; 6; 5; 4; 9]`.

2.11 `lfoldl` for left folding a list [7점]

(타입) `lfoldl: ('a * 'b -> 'b) -> 'b -> 'a list -> 'b`

(설명) `lfoldl` f e l takes e and the first item of l and applies f to them, then feeds the function with this result and the second argument and so on.

`lfoldl` f e $[x_1; x_2; ...; x_n]$ returns $f(x_n, ..., f(x_2, f(x_1, e))...)$ or e if the list is empty.

(실행 예) `lfoldl (fun (x, y) -> x - y) 0 [1; 2; 3]` returns 2.

2.12 `lzip` for pairing corresponding members of two lists [7점]

(타입) `lzip: 'a list -> b' list -> ('a * 'b) list`

(설명) `lzip` $[x_1; \cdots; x_n]$ $[y_1; \cdots; y_n]$ returns $[(x_1, y_1); \cdots; (x_n, y_n)]$.
If two lists differ in length, ignore surplus elements.

(실행 예) `lzip ["A"; "B"; "C"; "D"] [1; 2; 3; 4; 5; 6]` returns
`[("A", 1); ("B", 2); ("C", 3); ("D", 4)]`.

2.13 `split` for splitting a list into two lists [8점]

(타입) `split: 'a list -> 'a list * 'a list`

(설명) `split` l returns a pair of two lists. The first list consists of elements in odd positions and the second consists of elements in even positions in a given list respectively.
For an empty list, `split` returns ([ ], [ ]).

(실행 예) `split [1; 3; 5; 7; 9; 11]` returns `([1; 5; 9], [3; 7; 11])`.

(가정) 입력으로 주어지는 리스트의 길이는 짝수.

2.14 `cartprod` for the Cartesian product of two sets [8점]

(타입) `cartprod: 'a list -> 'b list -> ('a * 'b) list`

(설명) `cartprod S T` returns the set of all pairs (x, y) with $x \in S$ and $y \in T$.
The order of elements is important:
`cartprod` $[x_1; \cdots; x_n]$ $[y_1; \cdots; y_n]$ returns $[(x_1, y_1); \cdots; (x_1, y_n); (x_2, y_1); \cdots; (x_n, y_n)]$.

(실행 예) `cartprod [1; 2] [3; 4; 5]` returns `[(1, 3); (1, 4); (1, 5); (2, 3); (2, 4); (2, 5)]`.

3 테스트

작성한 프로그램을 OCaml 탑레벨 해석기에서 테스트하려면 다음과 같이 `#use` 커맨드를 이용하면 됩니다.

```
hsim@ubuntu:~/tmp/hw4$ ocaml
        OCaml version 4.02.3

# #use "hw4.ml";;
exception NotImplemented
val list_add : int list -> int list -> int list = <fun>
val insert : 'a -> 'a list -> 'a list = <fun>
val insort : 'a list -> 'a list = <fun>
val ltake : 'a list -> int -> 'a list = <fun>
val lall : ('a -> bool) -> 'a list -> bool = <fun>
...
# list_add [1; 2] [3; 4; 5];;
- : int list = [4; 6; 5]
```

4 제출

숙제 작성을 마친 후 반드시 `make`를 실행하여 숙제 프로그램이 컴파일이 잘 되는지 확인하세요. 컴파일이 안 되는 코드를 제출하면 본 숙제는 0점입니다.

```
hsim@ubuntu:~/tmp/hw4$ ls
hw4.ml  hw4.mli  Makefile
hsim@ubuntu:~/tmp/hw4$ make
ocamlc  -c hw4.mli -o hw4.cmi
ocamlc  -c hw4.ml -o hw4.cmo
ocamlc  -o hw4 hw4.cmo
```

끝으로 완성한 `hw4.ml` 파일을 과제 게시판에 업로드하면 됩니다.