2020 봄학기 프로그래밍언어
프로그래밍 과제 #3 (100점)

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

제출 마감: 5월 18일 월요일 11:59pm

1 주의사항

- 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 `hw3.zip` 파일을 다운받아 압축을 풉니다:

```
hsim@ubuntu:~/tmp$ unzip hw3.zip
Archive:  hw3.zip
   creating: hw3/
  inflating: hw3/.depend
  inflating: hw3/hw3.mli
  inflating: hw3/hw3.ml
  inflating: hw3/Makefile
```

반드시 `hw3.ml` 파일만 수정하세요. `hw3.ml` 파일은 다음과 같습니다:

```
exception Not_implemented

let rec sum _ = raise Not_implemented
let rec fac _ = raise Not_implemented
let rec fib _ = raise Not_implemented
let rec gcd _ = raise Not_implemented
let rec max _ = raise Not_implemented

type tree = Leaf of int | Node of int * tree * tree

let rec sum_tree _ = raise NotImplemented
let rec depth _ = raise NotImplemented
let rec bin_search _ _ = raise NotImplemented

  ... 이하 생략 ...
```

첫 번째 문제인 재귀 함수 `sum`을 작성하기 위해, 등호의 오른쪽에 있는 `raise Not_implemented`를 삭제하고 문제가 요구하는 올바른 함수 정의를 채워 넣습니다. 나머지 문제들도 동일한 방식으로 작성하면 됩니다.

2 문제

2.1 Recursive functions on integers

2.1.1 `sum` for adding integers 1 to n (inclusive) [10점]

(타입) `sum : int -> int`

(설명) `sum` n returns $\sum_{i=1}^{n} i$. 공식을 사용하지 말고 자기호출 함수로 작성할 것.

(가정) $n > 0$.

(실행 예)

```
# sum 10 ;;
- : int = 55
```

2.1.2 `fac` for factorials [10점]

(타입) `fac: int -> int`

(설명) `fac` n returns $\Pi_{i=1}^{n} i$. 즉, $1 \times \ldots \times n$을 자기호출 함수로 계산할 것.

(가정) $n > 0$.

(Hint) `sum` 함수를 조금만 변경하면 됨.

2.1.3 `fib` for Fibonacci numbers [10점]

(타입) `fib: int -> int`

(설명)
`fib` n returns `fib` $(n-1)$ + `fib` $(n-2)$ when $n \geq 2$.
`fib` n returns 1 if $n = 0$ or $n = 1$.

(가정) $n \geq 0$.

2.1.4 최대공약수를 계산하는 `gcd` 함수 [10점]

(타입) `gcd: int -> int -> int`

(설명) `gcd` m n는 유클리드 호제법을 이용하여 m과 n의 최대공약수를 반환한다.

(가정) $m \geq 0$, $n \geq 0$, $m + n > 0$.

(실행 예) 구현에 따라 계산 과정이 아래 예제와 다를 수 있음.

```
      gcd 15 20
↦     gcd 5 15
↦     gcd 0 5
↦     5
```

2.1.5 `max` for finding the largest integer in a list of integers [10점]

(타입) `max: int list -> int`

(설명) `max` l returns the largest integer in the list l. If an empty list is given, return 0.

(가정) Every element in a given list is greater than or equal to 0.

(실행 예) `max [5; 3; 6; 7; 4]` returns `7`.

2.2 Functions on binary trees

아래는 이진 나무 구조를 정의하는 타입 정의이다. 리프 노드(leaf node)도 정수 값을 인자로 가짐에 유의하라.

```
type tree = Leaf of int | Node of int * tree * tree
```

2.2.1 `sum_tree` for computing the sum of integers stored in a binary tree [10점]

(타입) `sum_tree : tree -> int`

(설명) `sum_tree` t returns the sum of integers stored in the tree t.

(실행 예) `sum_tree (Node (7, Node (3, Leaf 1, Leaf 2), Leaf 4))` returns `17`.

2.2.2 `depth` for computing the depth of a tree [10점]

(타입) `depth : tree -> int`

(설명) `depth` t returns the length of the longest path from the root to leaf.

(실행 예) `depth (Node (7, Node (3, Leaf 1, Leaf 2), Leaf 4))` returns `2`.

2.2.3 `bin_search` for searching an element in a binary search tree [10점]

(타입) `bin_search : tree -> int -> bool`

(설명) `bin_search` t x returns `true` if the number x is found in the binary search tree t; otherwise it returns `false`.

(가정) t는 이진 검색 나무: t의 모든 내부 노드(internal node) t'에 대해, t'의 왼쪽 하위 나무에 있는 노드들은 t'에 저장된 정수 값보다 작은 값을 가지며, t'의 오른쪽 하위 나무에 있는 노드들은 t'에 저장된 정수 값보다 큰 값을 갖는다. 또한 이진 검색 나무의 노드들에 저장된 정수 값들은 모두 다르다고 가정하라.

(실행 예) `bin_search (Node (4, Node (2, Leaf 1, Leaf 3), Leaf 7)) 2` returns `true`.
`bin_search (Node (4, Node (2, Leaf 1, Leaf 3), Leaf 7)) 5` returns `false`.

2.3 Arithmetic expressions

Consider the following type definition for arithmetic expressions.

```
type exp =
  INT of int
| ADD of exp * exp
| SUB of exp * exp
| MUL of exp * exp
| DIV of exp * exp
| MOD of exp * exp
```

2.3.1 `interp` for computing the value of an arithmetic expression [10점]

(타입) `interp : exp -> int`

(설명) `interp` e returns the integer value of the arithmetic expression e.

(실행 예) `interp (ADD (SUB (INT 100, INT 10), MUL (INT 2, INT 8)))` returns `106`.

2.4 Propositional logic

Consider the following type definition for propositional logic.

```
type formula =
  True
| False
| Neg of formula
| Or of formula * formula
| And of formula * formula
| Imply of formula * formula
```

2.4.1 `eval` for computing the boolean value of a formula [10점]

(타입) `eval : formula -> bool`

(설명) `eval` f returns the boolean value of the formula f.

(실행 예) `eval (Imply (And (True, Or (True, False)), False))` returns `false`.

3 테스트

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

```
hsim@ubuntu:~/tmp/hw3$ ocaml
        OCaml version 4.02.3

# #use "hw3.ml";;
exception NotImplemented
val sum : int -> int = <fun>
val fac : int -> int = <fun>
val fib : int -> int = <fun>
val gcd : int -> int -> int = <fun>
val max : int list -> int = <fun>
type tree = Leaf of int | Node of int * tree * tree
val sum_tree : tree -> int = <fun>
...
# sum 10;;
- : int = 55
# sum_tree (Node(3, Leaf 1, Leaf 2));;
- : int = 6
```

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/hw3$ ls
hw3.ml  hw3.mli  Makefile
hsim@ubuntu:~/tmp/hw3$ make
ocamlc  -c hw3.mli -o hw3.cmi
ocamlc  -c hw3.ml -o hw3.cmo
ocamlc  -o hw3 hw3.cmo
```

끝으로 완성한 `hw3.ml` 파일을 과제 게시판에 업로드하면 됩니다.