

Name:

Student Number:

2019 가을학기
Programming Languages 기말시험
Sample Solution

	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5	Total
Score						
Max	20	20	20	15	25	100

- 본 페이지 상단에 반드시 이름과 학번을 적으세요.
- 본 시험은 총 13 페이지 5문제로 구성되어 있습니다. 만점은 100점 입니다.
- 문제 2에서 답으로 작성한 함수는 OCaml 해석기에 그대로 입력한 것으로 간주됩니다. 따라서 작성한 코드에 사소한 오류가 있더라도 틀린 답으로 처리합니다. (단, 함수 정의 끝에 쓰는 `;;`는 생략하셔도 됩니다).
- 답을 읽기 쉽게 잘 적기 바랍니다. 담당 교수가 확인이 불가능한 답은 오답으로 처리합니다.
- 본 시험은 75분 동안 진행됩니다.

1 True of False [20점]

Question 1. [2점] 원소나열법을 이용하여 무한 집합을 정의할 수 있다.

false

Question 2. [2점] 다음은 명제 논리(propositional logic) 문법 정의이다.

$$A, B ::= T \mid F \mid \neg A \mid A \wedge B \mid A \vee B \mid A \Rightarrow B$$

$\llbracket (T \wedge (T \vee F)) \Rightarrow F \rrbracket$ 는 *false*를 의미한다.

true

Question 3. [2점] $n \geq b$ 를 만족하는 모든 자연수에 대해 수학적 귀납법을 이용하여 명제 $P(n)$ 이 성립함을 보이고자 한다. Base case로는 $P(b)$ 가 성립함을 보여야한다.

true

Question 4. [2점] C 언어는 다형 타입(polymorphic type)을 지원한다.

false

Question 5. [2점] C 언어는 고차 함수(higher-order function)을 지원한다.

false

Question 6. [2점] 다음과 같이 정의되는 이진 나무(binary tree) 구조를 고려하자.

$$t ::= \text{leaf} \mid \text{node}(n, t, t) \quad (n \in \mathbb{Z})$$

모든 이진 나무에서 leaf의 개수는 node의 개수보다 항상 정확하게 1개 더 많다.

true

Question 7. [2점] OCaml에서 다음 프로그램을 실행한 결과는 -5이다.

```
let x = 7 in
let y = 2 in
let y =
  let x = x - 1
  in x - y
in (x - 8) - y
```

true

Question 8. [2점] 다음과 같이 정의되는 람다 계산식(λ -calculus)만으로는 임의의 튜링 기계(Turing Machine)를 시뮬레이션 할 수 없다. 즉, 람다 계산식만으로는 현대 프로그래밍 언어로 작성 가능한 모든 프로그램을 작성할 수 없다.

$$e ::= x \mid \lambda x. e \mid e e$$

false

Question 9. [2점] 정적 유효범위(static scoping)는 변수의 유효범위(scope)가 프로그램을 컴파일할 때 결정되는 방식으로, 함수가 호출되어 함수의 몸체가 실행될 때, 함수 몸체에서 사용되는 자유 변수의 값을 결정하기 위해, 함수가 정의되는 시점에서의 실행 환경을 이용한다.

true

Question 10. [2점] Haskell과 같은 언어에서 사용하는 느긋한 계산(lazy evaluation) 방식에서는 함수 적용식(함수 호출식)을 계산할 때, 함수의 몸체에서 실질 인자의 값을 필요로 할 때까지 인자의 계산을 뒤로 미룬다.

true

2 OCaml 프로그래밍 [20점]

본 문제에서는 주어진 요구사항을 만족하는 다양한 함수를 구현합니다. 예를 들어 1부터 n 까지의 합을 계산하는 `sum` 함수는 다음과 같이 작성할 수 있습니다.

```
let rec sum n =  
  if n = 0 then 0  
  else n + sum (n - 1)
```

코드를 작성할 때, 함수 정의 끝에 쓰는 `;;`은 생략해도 됩니다.

Question 1. [3점] 정렬된 리스트에 원소를 삽입하는 `insert` 함수를 작성하라.

(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 (오름차순).

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

```
let rec insert a l =  
  match l with  
  | [] -> [a]  
  | x :: xs ->  
    if a < x then a :: l  
    else x :: insert a xs
```

Question 2. [3점] 위에서 정의한 `insert` 함수를 이용하여 삽입 정렬(insertion sort)를 구현하는 `insort` 함수를 작성하라.

(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]`.

```
let rec insort l =  
  match l with  
  | [] -> []  
  | x :: xs -> insert x (insort xs)
```

Question 3. [3점] 리스트를 입력받아 원소의 순서가 거꾸로된 리스트를 반환하는 `rev` 함수를 작성하라.

(타입) `rev : 'a list -> 'a list`

(설명) `rev l` reverses `l`.

(실행 예) `rev [0; 1; 2; 3; 4; 5]` returns `[5; 4; 3; 2; 1; 0]`.

```
let rec rev l =  
  match l with  
  | [] -> []  
  | x :: xs -> rev xs @ [x]
```

Question 4. [3점] 리스트와 조건(boolean 값을 반환하는 함수의 형태로, predicate)을 입력으로 받아 조건을 만족하는 원소들을 리스트로 반환하는 `filter` 함수를 작성하라.

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

(설명) `filter f l` returns every element of `l` that satisfies the predicate `f`.

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

```
let rec filter f l =  
  match l with  
  | [] -> []  
  | h :: t -> if f h then h :: filter f t else filter f t
```

Question 5. [4점] 이진 나무(binary tree)를 입력으로 받아 후위 순회(postorder traversal)를 수행하는 postorder 함수를 작성하라. 이진 나무의 타입은 다음과 같이 정의된다.

```
type 'a tree = Leaf of 'a | Node of 'a tree * 'a * 'a tree
```

(타입) postorder: 'a tree -> 'a list

(설명) postorder *t* returns a list of elements produced by a postorder traversal of the tree *t*.

(예제) postorder (Node (Node (Leaf 4, 2, Leaf 5), 1, Node (Leaf 6, 3, Leaf 7)))
returns [4; 5; 2; 6; 7; 3; 1].

```
let rec postorder t =  
  match t with  
  | Leaf x -> [x]  
  | Node (l, x, r) -> postorder l @ postorder r @ [x]
```

Question 6. [4점] 이진 탐색 나무(binary search tree)와 값을 입력으로 받아 이진 탐색 나무에 주어진 값이 있으면 true를 반환하고 그렇지 않으면 false를 반환하는 search 함수를 작성하라. 이진 탐색 나무의 타입은 다음과 같이 정의된다.

```
type 'a tree = Leaf | Node of 'a tree * 'a * 'a tree
```

```
let rec search t x =  
  match t with  
  | Leaf -> false  
  | Node (l, y, r) ->  
    if x = y then true  
    else if x < y then search l x  
    else search r x
```

3 함수와 변수의 유효범위 [20점]

다음은 \mathcal{L}^{rec} 의 추상 문법 구조이다.

<i>Expression</i>	<i>E</i> ::=	<i>n</i>	number
		<i>x</i>	variable
		<i>E</i> + <i>E</i>	addition
		<i>E</i> − <i>E</i>	subtraction
		iszero <i>E</i>	zero test
		if <i>E</i> then <i>E</i> else <i>E</i>	conditional expression
		let <i>x</i> = <i>E</i> in <i>E</i>	local variable declaration
		let rec <i>f x</i> = <i>E</i> in <i>E</i>	local recursive function declaration
		fun <i>x</i> -> <i>E</i>	anonymous function
		<i>E</i> <i>E</i>	function application

정적 유효범위(static scoping) 하에서 의미 공간(semantic domain)은 다음과 같이 정의된다.

$$\begin{aligned}
 Val &= \mathbb{Z} + Bool + Fun + RecFun \\
 Fun &= Var \times Exp \times Env \\
 RecFun &= Var \times Var \times Exp \times Env \\
 \rho \in Env &= Var \rightarrow Val
 \end{aligned}$$

실행 의미구조를 정의하기 위해 다음과 같은 계산 판단문(evaluation judgment)을 이용한다.

$$\rho \vdash e \Rightarrow v \iff \text{Under environment } \rho, \text{ expression } e \text{ evaluates to } v.$$

$\rho \vdash e \Rightarrow v$ 는 환경이 ρ 일 때, 표현식 e 가 값 v 로 계산됨을 의미한다.

Question 1. [8점] 위에서 정의한 계산 판단문을 이용하여 아래 계산 규칙(evaluation rules)들을 완성하라. 정적 유효범위를 가정하고, 정수는 n , 값은 v , 환경은 ρ 으로 표기하라 ($n \in \mathbb{Z}$, $v \in Val$, and $\rho \in Env$).

$$\frac{}{\rho \vdash \text{fun } x \rightarrow E \Rightarrow (x, E, \rho)} \qquad \frac{[f \mapsto (f, x, E_1, \rho)]\rho \vdash E_2 \Rightarrow v}{\rho \vdash \text{let rec } f \ x = E_1 \text{ in } E_2 \Rightarrow v}$$

Non-recursive function:

$$\frac{\rho \vdash E_1 \Rightarrow (x, E, \rho') \quad \rho \vdash E_2 \Rightarrow v \quad [x \mapsto v]\rho' \vdash E \Rightarrow v'}{\rho \vdash E_1 \ E_2 \Rightarrow v'}$$

Recursive function:

$$\frac{\rho \vdash E_1 \Rightarrow (f, x, E, \rho') \quad \rho \vdash E_2 \Rightarrow v \quad [x \mapsto v, f \mapsto (f, x, E, \rho')]\rho' \vdash E \Rightarrow v'}{\rho \vdash E_1 \ E_2 \Rightarrow v'}$$

Question 2. [4점] 동적 유효범위(dynamic scoping)를 가정하고 아래 계산 규칙들을 완성하라.

$$\overline{\rho \vdash \text{fun } x \rightarrow E \Rightarrow (x, E)}$$

$$\frac{\rho \vdash E_1 \Rightarrow (x, E) \quad \rho \vdash E_2 \Rightarrow v \quad [x \mapsto v]\rho \vdash E \Rightarrow v'}{\rho \vdash E_1 E_2 \Rightarrow v'}$$

Question 3. [4점] 아래 프로그램을 실행했을 때 최종 계산 결과를 적어라.

```
let a = 3 in
let f = fun z -> a in
let g = fun a -> f 0 in
let a = 5 in
g 2
```

• 정적범위: 3

• 동적범위: 2

Question 4. [2점] 양의 정수를 인자로 받아 두 배 값을 반환하는 double 함수를 완성하라. 반드시 \mathcal{L}^{rec} 의 문법만 이용해야 한다.

```
let rec double x =
  if iszero x then 0 else 2 + double (x - 1)
in double 6
```

Question 5. [2점] 다음 코드를 커링(currying)을 이용하여 표현하라.

Church-style:

```
let f = fun (x, y) -> x + y in f (3, 4)
```

Curry-style:

```
let f = fun x -> fun y -> x + y in f 3 4
```


4 이름 없는 표현식 [15점]

다음은 \mathcal{L}^{fun} 의 이름 없는 표현식(nameless expression)이다.

<i>Nameless Expression</i>	$N ::=$	n	
		$\#n$	lexical address
		$N + N$	
		$N - N$	
		iszero N	
		if N then N else N	
		let N in N	
		fun N	
		$N N$	

의미 공간(semantic domain)은 다음과 같이 정의된다.

$$\begin{aligned}
 \textit{Nameless Value} \quad NVal &= \mathbb{Z} + \textit{Bool} + NFun \\
 \textit{Nameless Closure} \quad NFun &= NExp \times NEnv \\
 \textit{Nameless Environment} \quad \varphi \in NEnv &= NVal \textit{ list}
 \end{aligned}$$

$v :: \varphi$ 는 환경 φ 의 맨 앞에 값 v 를 추가하는 연산이고, $nth(\varphi, n)$ 은 환경 φ 의 n 번째 원소를 반환하는 연산이다. 환경의 인덱스(index)는 0부터 시작한다.

실행 의미구조를 정의하기 위해 다음과 같은 계산 판단문(evaluation judgment)을 이용한다.

$$\varphi \vdash e \Rightarrow v \iff \text{Under environment } \varphi, \text{ expression } e \text{ evaluates to } v.$$

$\varphi \vdash e \Rightarrow v$ 는 환경이 φ 일 때, 표현식 e 가 값 v 로 계산됨을 의미한다.

Question 1. [10점] 위에서 정의한 계산 판단문을 이용하여 아래 계산 규칙(evaluation rules)들을 완성하라. 정적 유효범위를 가정하고, 정수는 n , 값은 v , 환경은 φ 으로 표기하라 ($n \in \mathbb{Z}$, $v \in Val$, and $\varphi \in NEnv$). 정수를 위한 계산 규칙은 이미 완성되어 있다.

$$\begin{array}{c}
 \frac{}{\varphi \vdash n \Rightarrow n} \qquad \frac{}{\varphi \vdash \#n \Rightarrow nth(\varphi, n)} \\
 \\
 \frac{\varphi \vdash N_1 \Rightarrow n_1 \quad \varphi \vdash N_2 \Rightarrow n_2}{\varphi \vdash N_1 + N_2 \Rightarrow n_1 + n_2} \\
 \\
 \frac{\varphi \vdash N_1 \Rightarrow v_1 \quad v_1 :: \varphi \vdash N_2 \Rightarrow v}{\varphi \vdash \text{let } N_1 \text{ in } N_2 \Rightarrow v} \\
 \\
 \frac{}{\varphi \vdash \text{fun } N \Rightarrow (N, \varphi)} \\
 \\
 \frac{\varphi \vdash N_1 \Rightarrow (N, \varphi') \quad \varphi \vdash N_2 \Rightarrow v \quad v :: \varphi' \vdash N \Rightarrow v'}{\varphi \vdash N_1 N_2 \Rightarrow v'}
 \end{array}$$

Question 2. [5점] 위에서 정의한 계산 규칙을 이용하여 다음 판단문을 증명하라. (부분 점수 없음, 완벽하게 작성해야 정답으로 인정)

$\square \vdash (\text{let } 37 \text{ in fun (let (\#0 - \#1) in (\#2 - \#1))) 10 \Rightarrow 27$

5 변경 가능한 상태 [25점]

다음은 암시적 참조를 지원하는 \mathcal{L}_{imp}^{ref} 의 추상 문법 구조이다.

$$\begin{array}{lcl} \text{Expression } E & ::= & n \mid x \mid E + E \mid E - E \\ & & \mid \text{iszero } E \mid \text{if } E \text{ then } E \text{ else } E \\ & & \mid \text{let } x = E \text{ in } E \\ & & \mid \text{fun } x \rightarrow E \mid E \ E \mid E \langle y \rangle \\ & & \mid \text{set } x = E \\ & & \mid E; E \end{array}$$

$E \ E$ 는 값에 의한 호출(call-by-value)이고 $E \langle y \rangle$ 는 참조에 의한 호출(call-by-reference)이다. $\text{set } x = E$ 는 변수 x 의 값을 E 를 계산한 결과로 업데이트하는 연산이며, $E_1; E_2$ 는 표현식을 순서대로 계산하는 sequence expression이다.

한편, 의미 공간(semantic domain)은 다음과 같이 정의된다.

$$\begin{array}{lcl} \text{Val} & = & \mathbb{Z} + \text{Bool} + \text{Closure} \\ \text{Closure} & = & \text{Var} \times \text{Exp} \times \text{Env} \\ \rho \in \text{Env} & = & \text{Var} \rightarrow \text{Loc} \\ \sigma \in \text{Mem} & = & \text{Loc} \rightarrow \text{Val} \end{array}$$

실행 의미구조를 정의하기 위해 다음과 같은 계산 판단문(evaluation judgment)을 이용한다.

$$\rho, \sigma \vdash e \Rightarrow v, \sigma' \iff \text{Under environment } \rho \text{ and memory } \sigma, \text{ expression } e \text{ evaluates to } v \text{ and the memory } \sigma \text{ is updated to } \sigma'.$$

$\rho, \sigma \vdash e \Rightarrow v, \sigma'$ 은 환경이 ρ 이고 메모리가 σ 일 때, 표현식 e 가 값 v 로 계산되고 메모리 σ 는 σ' 으로 업데이트됨을 의미한다.

Question 1. [12점] 위에서 정의한 계산 판단문을 이용하여 아래 계산 규칙(evaluation rules)들을 완성하라. 정적 유효범위를 가정하고, 정수는 n , 값은 v , 환경은 ρ , 메모리는 σ 으로 표기하라 ($n \in \mathbb{Z}$, $v \in \text{Val}$, $\rho \in \text{Env}$, and $\sigma \in \text{Mem}$).

$$\begin{array}{c} \frac{}{\rho, \sigma \vdash x \Rightarrow \sigma(\rho(x)), \sigma} \qquad \frac{\rho, \sigma_0 \vdash E \Rightarrow v, \sigma_1}{\rho, \sigma_0 \vdash \text{set } x = E \Rightarrow v, [\rho(x) \mapsto v]\sigma_1} \\[10pt] \frac{\rho, \sigma_0 \vdash E_1 \Rightarrow v_1, \sigma_1 \quad [x \mapsto l]\rho, [l \mapsto v_1]\sigma_1 \vdash E_2 \Rightarrow v, \sigma_2}{\rho, \sigma_0 \vdash \text{let } x = E_1 \text{ in } E_2 \Rightarrow v, \sigma_2} \quad (l \notin \text{Dom}(\sigma_1)) \\[10pt] \frac{\rho, \sigma_0 \vdash E_1 \Rightarrow (x, E, \rho'), \sigma_1 \quad \rho, \sigma_1 \vdash E_2 \Rightarrow v, \sigma_2 \quad [x \mapsto l]\rho', [l \mapsto v]\sigma_2 \vdash E \Rightarrow v', \sigma_3}{\rho, \sigma_0 \vdash E_1 \ E_2 \Rightarrow v', \sigma_3} \quad (l \notin \text{Dom}(\sigma_2)) \\[10pt] \frac{\rho, \sigma_0 \vdash E_1 \Rightarrow (x, E, \rho'), \sigma_1 \quad [x \mapsto \rho(y)]\rho', \sigma_1 \vdash E \Rightarrow v', \sigma_2}{\rho, \sigma_0 \vdash E_1 \langle y \rangle \Rightarrow v', \sigma_2} \\[10pt] \frac{\rho, \sigma_0 \vdash E_1 \Rightarrow v_1, \sigma_1 \quad \rho, \sigma_1 \vdash E_2 \Rightarrow v_2, \sigma_2}{\rho, \sigma_0 \vdash E_1; E_2 \Rightarrow v_2, \sigma_2} \end{array}$$

Question 2. [8점] 아래 프로그램들을 실행했을 때 최종 계산 결과를 적어라.

```
let p = fun x -> (set x = 4) in
let a = 3 in
(p a; a)
```

3

```
let p = fun x -> (set x = 4) in
let a = 3 in
(p <a>; a)
```

4

```
let f = fun x -> (set x = 44) in
let g = fun y -> f <y> in
let z = 55 in
(g <z>; z)
```

44

```
let b = 3 in
let p = fun x -> fun y -> (set x = 4; y) in
(p <b>) <b>
```

4

Question 3. [5점] 다음은 명시적 참조를 지원하는 \mathcal{L}_{exp}^{ref} 의 추상 문법 구조이다.

<i>Expression</i>	<i>E</i>	$::=$	$n \mid x \mid E + E \mid E - E$	
			$\mid \text{iszero } E \mid \text{if } E \text{ then } E \text{ else } E$	
			$\mid \text{let } x = E \text{ in } E$	
			$\mid \text{fun } x \rightarrow E \mid E E$	
			$\mid \text{ref } E$	memory allocation
			$\mid ! E$	memory dereference
			$\mid E := E$	memory update
			$\mid E; E$	sequence expression

아래 프로그램들을 실행했을 때 최종 계산 결과를 적어라.

```
let counter = ref 0 in
let f = fun x -> (counter := !counter + 1; !counter) in
let a = f 0 in
let b = f 0 in
b - a
```

1

```
let f =
  let counter = ref 0 in
  fun x -> (counter := !counter + 1; !counter)
in
let a = f 0 in
let b = f 0 in
a - b
```

-1

```
let f =
  fun x ->
    let counter = ref 0 in
    (counter := !counter + 1; !counter)
in
let a = f 0 in
let b = f 0 in
b - a
```

0