

Example Solutions for Programming Languages G22.2110 Summer 2007 Midterm

6/21/2007

These are example solutions. Please keep in mind that often, there is not just one correct solution to a question. If you come up with different answers, then it may be that both your answers and these answers here are correct. Of course, these answers here may also contain mistakes; if you spot some, please let us know so we can correct them.

1. Syntax (20 = 10 + 10 points)

Consider the following EBNF grammar with start symbol $\langle s \rangle$:

$\langle s \rangle ::= "*" \langle s \rangle \mid \langle s \rangle "[" \langle s \rangle "]" \mid \langle id \rangle$
 $\langle id \rangle ::= "a" \mid "b" \mid \dots \mid "z"$

1a. (10 points) Show both parse trees for the expression $*p[i]$.

Example solutions

1b. (10 points) Change the grammar so $"[...]"$ has higher precedence than $"*"$.

Example solutions

$\langle s \rangle ::= "*" \langle s \rangle \mid \langle t \rangle$
 $\langle t \rangle ::= \langle t \rangle "[" \langle s \rangle "]" \mid \langle id \rangle$
 $\langle id \rangle ::= "a" \mid "b" \mid \dots \mid "z"$

2. Functional rewriting (20 = 10 + 10 points)

2a. (10 points) Consider the following Scheme function:

```
(define (a N L) (if (null? L) L (cons (+ N (car L)) (a N (cdr L)))))
```

Show how Scheme evaluates the expression `(cons 2 (a 4 '()))` by rewriting it.

Example solutions

```
(cons 2 (a 4 '()))  
=> (cons 2 (if (null? '()) '() (cons (+ 4 (car '())) (a 4 (cdr '())))))  
=> (cons 2 (if #t '() (cons (+ 4 (car '())) (a 4 (cdr '())))))  
=> (cons 2 '())  
=> '(2)
```

2b. (10 points) Consider the following Scheme function:

```
(define (add-or-mul op x y) ((if (eq? "add" op) + *) x y))
```

Show how Scheme evaluates the expression `(add-or-mul "mul" -2 3)` by rewriting it.

Example solutions

```
(add-or-mul "mul" -2 3)  
=> ((if (eq? "add" "mul") + *) -2 3)  
=> ((if #f + *) -2 3)  
=> (* -2 3)  
=> -6
```

3. Recursion (20 = 5 + 5 + 5 + 5 points)

3a. (5 points) Briefly define “mutual recursion”.

Example solutions

Mutual recursion is when two or more entities refer to each other in a circle.

3b. (5 points) What is the difference between `let`, `let*`, and `letrec` in Scheme?

Example solutions

All names bound by `let` become available after the last definition, each name bound by `let*` becomes available after its own definition, and all names bound by `letrec` become available before the first definition.

3c. (5 points) Give an example for mutually recursive functions in Scheme.

Example solutions

```
(letrec ((ping (lambda (n) (if (= 0 n) "e\n" (pong (- n 1)))))
         (pong (lambda (n) (if (= 0 n) "o\n" (ping (- n 1)))))
        ...))
```

3d. (5 points) Give an example for mutually recursive types in C.

Example solutions

```
struct s;
struct t { struct s* ps; };
struct s { struct t* pt; };
```

4. Scoping ($N = N + N + N$ points)

Consider the following Python program:

```
def f(x):
    a = x
    def g():
        def h():
            print a
            h()
        def i(z):
            a = z
            g()
        i("f/i")
    f("main/f")
```

4a. (5 points) What does this program print, using static scoping?

Example solutions

```
main/f
```

4b. (15 points) Show the frames on the stack when `h` has just been called. For each frame, show the static and dynamic links.

Example solutions

ADR	SUBR	VARNAME	VALUE

86	h	static link	88
87		dynamic link	88

88	g	static link	94
89		dynamic link	90

```

-----
90 i   z           "f/i"
91     a           "f/i"
92     static link 94
93     dynamic link 94
-----
94 f   x           "main/f"
95     a           "main/f"
96     static link 98
97     dynamic link 98
-----
98 main static link -
99     dynamic link -

```

5. Scheme (20 = 6 + 14 points)

5a. (6 points) Write a Scheme function “make-pair” that takes two arguments, and returns a list that contains these two arguments. For example,

```

(make-pair 2 3) ⇒ (2 3)
(make-pair "hello" "world") ⇒ ("hello" "world")

```

Example solutions

```

(define (make-pair x y) (cons x (cons y '())))

```

5b. (14 points) Write a Scheme function “zip-lists” that takes two list arguments, and returns a list of pairs: the pair of the first element of both lists, the pair of the second element of both lists, etc. For example,

```

(zip-lists '(1 2) '(3 4)) ⇒ ((1 3) (2 4))
(zip-lists '(97 98 99) '("a" "b" "c")) ⇒ ((97 "a") (98 "b") (99 "c"))

```

You can assume that the input is always well-formed, i.e., both arguments are lists, and they have the same length.

Example solutions

```

(define (zip-lists x y)
  (if (null? x)
      x
      (cons
        (make-pair (car x) (car y))
        (zip-lists (cdr x) (cdr y)))))

```

<http://www.cs.nyu.edu/courses/summer07/G22.2110-001/midterm-example-solutions.pdf>

Total points: 100.