

Example Solutions for  
Programming Languages G22.2110 Summer 2007  
Final Exam

8/2/2007

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. Pointer arithmetic (20 = 10 · 2 points)  
Consider the following C variable declarations:

```
long long p[3][3];  
long long (*q)[3] = &p[2];  
long long *r = &p[2][2];
```

Assume that the array `p` starts at address 1000, and that `sizeof(long long) == 8` and `sizeof(int) == 4`. What are the numeric values of each of the following expressions?

- 1a. (2 points) `q`
- 1b. (2 points) `q + 1`
- 1c. (2 points) `q - 1`
- 1d. (2 points) `q - p`
- 1e. (2 points) `(long long*)q - (long long*)p`
- 1f. (2 points) `(int*)q - (int*)p`
- 1g. (2 points) `r`
- 1h. (2 points) `r + 1`
- 1i. (2 points) `r - &p[2][0]`
- 1j. (2 points) `r - &p[1][0]`

*Example solutions*

```
1a. q == &p[2] == 1000 + 2 * sizeof((long long)[3]) == 1000 + 2 * 3 * 8  
    == 1048
```

```

1b. q + 1 == 1048 + sizeof((long long)[3]) == 1048 + 3 * 8
    == 1072
1c. q - 1 == 1048 - sizeof((long long)[3]) == 1048 - 3 * 8
    == 1024
1d. q - p == (1048 - 1000) / sizeof((long long)[3]) == 48 / (3 * 8)
    == 2
1e. (long long*)q - (long long*)p == (1048 - 1000) / sizeof(long long) == 48 / 8
    == 6
1f. (int*)q - (int*)p == (1048 - 1000) / sizeof(int) == 48 / 4
    == 12
1g. r == &(p[2][2])
    == 1000 + 2 * sizeof((long long)[3]) + 2 * sizeof(long long)
    == 1000 + 2 * 3 * 8 + 2 * 8 == 1000 + 48 + 16
    == 1064
1h. r + 1 == 1064 + sizeof(long long) == 1064 + 8
    == 1072
1i. r - &p[2][0] == (1064 - 1048) / sizeof(long long) == 16 / 8
    == 2
1j. r - &p[1][0]
    == (1064 - (1000 + sizeof((long long)[3]))) / sizeof(long long)
    == (1064 - (1000 + 3 * 8)) / 8 == (1064 - 1000 - 24) / 8 == 40 / 8
    == 5

```

2. Type equivalence and compatibility (20 = 5 · 4 points)

2a. (4 points) Give an example of two types that are structurally equivalent but not name equivalent.

*Example solutions*

```

struct A { int x; struct A* y; };
struct B { int x; struct B* y; };

```

2b. (4 points) Give an example of two types that are not structurally equivalent.

*Example solutions*

```

struct C { int x; struct C* y; };
struct D { int x; int y; };

```

2c. (4 points) Give an example of two types such that one is compatible with the other, but they are not equivalent.

*Example solutions*

```

class A { }
class B extends A { void f(); } /* B is compatible with A. */

```

2d. (4 points) Is type compatibility symmetric? In other words, if type A is compatible with type B, does that imply that type B is also compatible with type A? Briefly explain your answer.

Example solutions

No. For example, in Java, type `String` is a subclass of type `Object`, so a value of type `String` is accepted wherever a value of type `Object` is expected. But in general, a value of type `Object` is not accepted wherever a value of type `String` is expected. For example, the following code yields a compiler error:

```
Object o = new Object();
String s = o;
```

Another way to see this is that `A` is compatible with `B` if `A` is a subset of `B`, but that doesn't imply that `B` is a subset of `A`.

- 2e. (4 points) Is type compatibility transitive? In other words, if type `A` is compatible with type `B`, and type `B` is compatible with type `C`, does that imply that type `A` is also compatible with type `C`? Briefly explain your answer.

Example solutions

Yes. If a value of type `B` is accepted wherever a value of type `C` is expected and a value of type `A` is accepted wherever a value of type `B` is expected, then a value of type `A` is accepted wherever a value of type `C` is expected. Another way to see this is that `A` is a subset of `B` and `B` a subset of `C`, hence `A` is a subset of `C`.

3. Virtual method dispatch (20 = 5 · 4 points)

Consider the following Java definitions:

```
interface I { public void a(); public void c(); }
class S { public void a() {} void b() {} }
class T extends S { public void c() {} }
class U extends T implements I { public void a() {} }
class V extends S { public void a() {} void d() {} }
class W extends V implements I { public void c() {} }
```

- 3a. (4 points) What is the vtable of class `S`?

Example solutions

```
0: S::a
1: S::b
```

- 3b. (4 points) What is the vtable of class `T`?

Example solutions

```
0: S::a
1: S::b
2: T::c
```

- 3c. (4 points) What is the vtable of class `U`?

Example solutions

0: U::a

1: S::b

2: T::c

--- note: this vtable doesn't assist interface dispatch, but it works as usual for dispatching through class-typed pointer

3d. (4 points) What is the vtable of class V?

Example solutions

0: V::a

1: S::b

2: V::d

3e. (4 points) What is the vtable of class W?

Example solutions

0: V::a

1: S::b

2: V::d

3: W::c

--- note: this vtable doesn't assist interface dispatch, but it works as usual for dispatching through class-typed pointer

4. Type inference (20 = 3 + 3 + 3 + 11 points)

Consider the following SML function:

```
fun q (_, x, 1) = x | q (s, y, n) = s (y, q (s, y, n - 1));
```

4a. (3 points) What is the result of “q (op \*, 2.0, 4);”?

Example solutions

q (op \*, 2.0, 4)

=> 2.0 \* q(op \*, 2.0, 3)

=> 2.0 \* 2.0 \* q(op \*, 2.0, 2)

=> 2.0 \* 2.0 \* 2.0 \* q(op \*, 2.0, 1)

=> 2.0 \* 2.0 \* 2.0 \* 2.0

=> 16.0

4b. (3 points) What is the result of “q (op +, 2.0, 4);”?

Example solutions

q (op +, 2.0, 4)

=> 2.0 + q(op +, 2.0, 3)

=> 2.0 + 2.0 + q(op +, 2.0, 2)

=> 2.0 + 2.0 + 2.0 + q(op +, 2.0, 1)

=> 2.0 + 2.0 + 2.0 + 2.0

=> 8.0

4c. (3 points) What is the type of q?

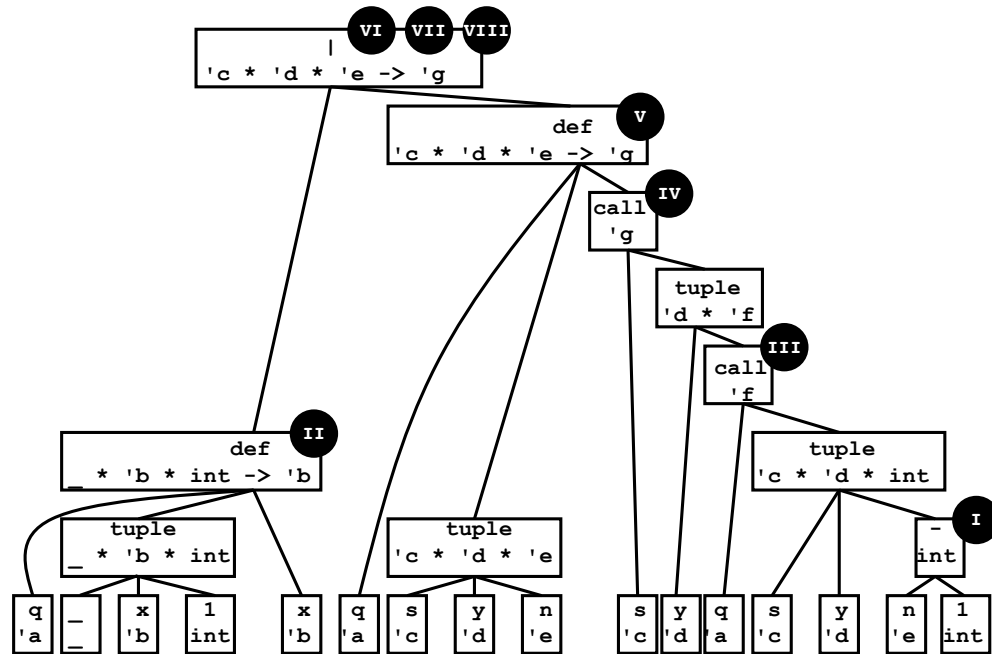
Example solutions

`('a * 'a -> 'a) * 'a * int -> 'a`

4d. (11 points) Show the type inference steps for q.

Example solutions

Drawing the definition as a tree, assigning types to leaves, propagating types up to internal nodes:



```
fun q (_, x, 1) = x | q (s, y, n) = s (y, q (s, y, n - 1));
```

Constraints:

- VI 'b = 'd
- VII int = 'e
- VIII 'b = 'g
- II 'a = \_ \* 'b \* int -> 'b
- III 'a = 'c \* 'd \* int -> 'f
- IV 'c = 'd \* 'f -> 'g
- V 'a = 'c \* 'd \* 'e -> 'g
- I 'e = int

Unification: the type of q is 'a, which is

```
'a = 'c * 'd * 'e -> 'g // constraint V
    = ('d * 'f -> 'g) * 'd * 'e -> 'g // constraint IV
    = ('d * 'f -> 'g) * 'd * int -> 'g // constraint VII
    = ('b * 'f -> 'g) * 'b * int -> 'g // constraint VI
    = ('g * 'f -> 'g) * 'g * int -> 'g // constraint VIII
    = ('g * 'g -> 'g) * 'g * int -> 'g // constraint III
```

Result:

```
q : ('g * 'g -> 'g) * 'g * int -> 'g
```

5. Parameter passing modes (20 = 4 · 5 points)  
Consider the following program in pseudo-C:

```
char x[3];
int y;
void f(char@ z) {
    y--;
    z = 'd';
    x[2] = x[1];
}
int main() {
    x[0] = 'a', x[1] = 'b', x[2] = 'c';
    y = 1;
    f(x[y]);
    printf("%c %c %c\n", x[0], x[1], x[2]);
}
```

- 5a. (5 points) What does the program print if *z* is passed by value?

Example solutions  
a b b

- 5b. (5 points) What does the program print if *z* is passed by value-result?

Example solutions  
a d b

- 5c. (5 points) What does the program print if *z* is passed by reference?

Example solutions  
a d d

- 5d. (5 points) What does the program print if *z* is passed by name?

Example solutions  
d b b

---

<http://www.cs.nyu.edu/courses/summer07/G22.2110-001/final-example-solutions.pdf>  
Total points: 100.