

# CS 6V81-002: Quiz 7 Solutions

February 18, 2008

*Students did somewhat poorly on this quiz, so I've included some extended commentary below explaining the correct solutions.*

1. Cyclone enforces which of the following security policies? (circle all that apply)

**(a) control-flow integrity**

**(b) memory safety**

(c) all memory locations must be written to before they can be read

(d) all allocated memory regions must eventually be freed

*Cyclone programs cannot branch outside of the control-flow graph defined by the static program structure, and they cannot reference unallocated memory, so that means that both control-flow integrity and memory safety are enforced by the language design. Pointers cannot be read before they are written to (in order to prevent memory safety violations) but other data like integers can be used prior to initialization, so (c) is not correct. Allocated memory regions might never be freed since programs can be non-terminating, so (d) is not correct either.*

2. Suppose a Cyclone program declares `p` to be a “fat pointer” (e.g., `int ?p;`). Which of the following operations on `p` will cause the compiler to insert a runtime bounds check? (circle all that apply)

**(a) dereferences of `p` (e.g., `*p=5`)**

(b) pointer arithmetic on `p` (e.g., `p=p+1`)

**(c) casting `p` to a regular pointer (e.g., `(int *)p`)**

**(d) casting `p` to a never-null pointer (e.g., `(int @)p`)**

*Fat pointers are checked at dereference time, so (a) is correct but (b) is not. Regular pointers must always point either to `NULL` or to a writable memory address, so casting a fat pointer to a regular pointer requires a bounds check. Similarly, never-null pointers always point to a writable address, so (d) requires a bounds-check as well.*

3. Which of the following are true of “regions” in Cyclone? (circle all that apply)

(a) A region created by one function can be freed by another function.

**(b) Region variables exist only at compile-time; they have no runtime representation.**

(c) Region variables never alias.

**(d) Any attempt to store into a freed region is caught at compile-time.**

Regions are always freed at the end of the scope that created them, so (a) is incorrect. Region variables (e.g. `'r`) are type variables that don't exist at runtime. (However, region handles are runtime variables, so if you got the two confused, you probably got (b) wrong.) Region variables do sometimes alias. For example, if function `foo` expects two pointers as arguments, one in region `'r` and one in region `'s`, then Cyclone has no way to know whether at runtime the two pointer arguments might point into the same region, causing `'r` and `'s` to alias. Finally, Cyclone rejects any program that allows a pointer to escape the scope that defines the region it points into, making (d) a correct answer.

4. Cyclone replaces unions with tagged-unions because... (circle one)
- (a) the tags are required by Cyclone's safe memory manager
  - (b) the tags prevent integer fields from being used as pointers**
  - (c) the tags allow the programmer to prove to the compiler that certain runtime checks are unnecessary
  - (d) the tags allow the compiler to infer that certain runtime checks are unnecessary without requiring assistance from the programmer

*The point of tagged unions is to prevent non-pointer fields in a union from being used as pointers. For example, in C one can violate memory safety by writing the following:*

```
union { int i; int *p; } x;
x.i = 0xBAD;
*p = 0;
```

*causing the memory at address 0xBAD to be dereferenced. This is not possible with a tagged union since tagged unions only allow you to read from the field you last wrote to.*

5. Cyclone implicitly casts array variables (e.g., `"int a[64];"`) to fat pointers or to never-null pointers as necessary. For example, the operation `a[i]` (where `i` is an integer variable) would implicitly cast `a` to a fat pointer. Give an example of a command or expression that would cause `a` to be implicitly cast to a never-null pointer. Do not use any explicit casts or never-null pointer variables in your example (e.g., do not use the trivial examples `"(int @)a"` or `"int @p=a;"`).

*Any operation that dereferences `a` without doing pointer arithmetic would work, such as: `*a` and `a[0]`. Tests of the nullity of `a` also work, such as: `if (a) i=0;`.*