isl homework

1. Which of these sets are bounded (have a finite number of elements)?
    1. {i : i > 0}
    2. {i : i % 2 = 0}
    3. {floor(i / floor(i / 2)) : i > 0}
2. What do I mean when I say that a map’s space is wrapped and a set’s space isn’t? {[i0]} and {[[i0] -> [i0]]} are examples.
3. What kind of operation you would need to transform the set {A[i0]} into the map {A[i0] -> A[i0]}. isl::mapFromDomainAndRange() is one easy way to do it. Hint: isl::Map::identity takes a space argument.
4. Are the puma (isl::PwUnionMultiAff) and mupa (isl::UnionPwMultiAff) equivalent? Can you tell which is which? Can you think of a case where a mupa is not convertible into a puma? {((i) : i > 0), ((i - 1), i > 0} and {((i, i - 1), i > 0}
5. A[2*i + 3].B[8*i + 3*j - 7] can be represented by the access A_B[A[2*i + 3] -> B[8*i + 3*j - 7]], and one would think that it’s possible to decompose any affine matrix or struct access in this manner. Do you think it’s possible to represent recursive data structures like this?
6. In polly, we slurp up some programming language ast, delete the SCoP, and recreate it from scratch. You’d imagine that we have to reuse Vars for readability, but how do we keep track of them? (Hint: read the next question).
7. Most isl data structures can have an isl::Id attached to them to name them. So, in {A[i0]}, ‘A’ is the “name” part of the Id. There’s another clever functionality: you can attach a “user object” to it by going to and coming out of `void*`. What kind of cast is appropriate here?
8. isl is a C library that we have wrapped using a Python-script-generated C++ wrapper, and this does memory management for us. If you have a `isl_set* isl_set_foo(isl_set* aSet, isl_set* aOther)`, you’d naturally create an isl::Set class and make the first argument in the function call `this`. What do you do if `this` is consumed by isl (it often is, and is annotated with _isl__take)? Would you let the user have the terrible experience of calling `set.foo()` and then have `set` become an invalid object?
9. The convention in isl is never to modify anything that is passed, except to consume it. The `set = set.foo()` idiom is common, so `set.foo()` is always a bug. Would you be able to catch this bug at compile-time?
10. What kind of space can isl::UnionMap::getSpace() possibly return? Remember that there are no union-spaces.