Homework 4

Problem 1

a)

Num is a supertype of Fractional, Int, Integer, and Float

Fractional is a supertype of Float and a subtype of Num

Int and Integer are subtypes of Num

Float is a supertype of Int, Integer, and Num, and a subtype of Fractional.

b)

Div and Mod belong to the same type class, while + is different.

Running :t div, :t mod, and :t + we get:

- div :: Integral a ⇒ a→a→a
- mod :: Integral a ⇒ a→a→a
- (+) :: Num a ⇒ a→a→a

Div and Mod are of type Integral which can only operate on integers, while + is a Num type which can include floating point values.

c)

Float and ints are primitive types in C++, they are not subtypes or supertypes of each other.

Const float and Const int aren't a sub-type of the earlier, respectively, however they are given the constraint that their values cannot be modified after initialization.

Problem 2

a)

Dynamically typed because the declaration of the variables don’t have type declarations, indicating that the language determines typing at run-time.

b)

The scoping strategy relies on the position of the variable in the block of code, similar to that of C and C++. Variables called outside the scope of a function or implementation block cannot be accessed.

c)

The scoping strategy is similar in its usage of lexical scoping, whether the scope of the variable is determined by the position in the code. If a variable is overridden in the scope of a block but not outside its block where it’s displayed, it will change behavior depending on when you choose to use the variable.

d)

Both n1 and n2 refer to the same object so they evaluate true when n1 == n2

s1 and s2 are assigned the same strings but they have different locations in memory. But since they have the same value, they evaluate to true.

This means that the == operator handles the evaluation by its value and not its memory location. This is similar to python I believe.

Problem 3

a)

```
def nth_fibonacci(n: int) -> float:
  phi = (1 + sqrt(5))/2
  psi = (10sqrt(5))/2
  return (phi**n - psi**n)/sqrt(5)
```

c)

No, add :: Num → Num → Num suggests that both a and b must be numeric and should return a numeric type, but it could be either-or float or integer. If we want the same type to be for both a, b, and the return it would be better to use add :: (Num a) ⇒ a → a → a

Problem 4

a)

Since w.f is not initialized, the compiler seems to giving some number to avoid undefined behavior.

If a type doesn’t match the declared type given by a union, then the C++ compiler must handle the value itself.

b)

The error message seems to indicate that only one of the union fields can be acive at a time, resulting in an access to an inactive field and the given runtime error.

This seems to be safer than C++’s union type as C++ risks undefined behavior caused by both fields being active at the same time, while Zig provides a better optimization for memory and more flexibility.

Problem 5

a)

Each time the function is called, a new dictionary object res is created an returned with local variables name and res. These variables are destroyed after the end of the function, and thus have the same scope and lifetime as each other.

b)

The lifeime of x is limited to the scope of the inner block. When the block ends, x is destroyed meaning its location in memory is also destroyed. Therefore the the pointer n tries to point to the location, its attempt to dereference the pointer results in an undefined behavior.