Question 1

(a) see accompanying code

(b) 9

(c) 5

Question 2

(a)
- objects: 1,2
- homsets:
  (a) Hom(1,1): $\{id_1\}$
  (b) Hom(1,2), $\{f\}$
  (c) Hom(2,1), $\emptyset$
  (d) Hom(2,2), id_2
- Composition rules:
  (a) $id_1; f = f$
  (b) $id_1; id_1 = id_1$
  (c) $f; id_2 = f$
  (d) $id_2; id_2 = id_2$
- ids: id_1, id_2

(b)
- Unity: Covered by composition rules a, c
- Associativity: For all valid tripplets of composed morphisms, anything containing an identity automatically associates, by virtue of the composition rules 1 and 3. In this category, every morphism tripplet must contain at least one identity morphism. Therefore this category's morphisms are associative.
- we could also do a proof by cases, enumerating all triplets and showing their equivalence using the composition rules to show equivalence:

term 1	**term 2**	**both reduce to**	**composition rules used**
$(id_1; id_1); id_1$	$id_1; (id_1; id_1)$	id_1	b
$(id_2; id_2); id_2$	$id_2; (id_2; id_2)$	id_2	d
$(id_1; f); id_2$	$id_1; (f; id_2)$	f	a, c
$(f; id_2); id_2$	$f; (id_2; id_2)$	f	c, d
$(id_1; id_1); f$	$id_1; (id_1; f)$	f	a, b

Question 3

Yes, assuming $f; g$ and $g; f$ are composable as explicitly given by a composition rule then $f; g = id_c$ and $g; f = id_d$. Taken together, the former reduces to $c- > c$ and the latter to $d- > d$, which is equivalent to the identity operators id_c and id_d.

Details that I missed: $Hom(c, c)$ is defined to only contain the identity morphism. Therefore anything that reduces to $c- > c$ is equivalent to this set, and is by definition the identity morphism. The definition of "isomorphism" means $f; g = id_c \wedge g; f = id_d$, so showing the reduction of both to the identities proves the isomorphism

Question 4

a Have a monoid. For simplicity, assume there are two morphisms, f and g. define the composition law so that $f; g = g$, $g; f = g$, $f; f = g$, $g; g = g$. This ensures that associativity is always respected, as any string of compositions will always result in g. While unitivity is satisfied for g by the rules $f; g = g$ and $g; g = g$, it is not for f, since there is no rule $x; f = f$ for $x \in f, g$ is this possible to do for non-monoids?

b Monoid where morphisms are the natural numbers, identity is the number 1, and composition is exponentiation.

Question 5

For a,b since the object, morphisms, and composition rules are defined for us, the only thing we have to do is show that the compositions satisfy unitivity and associativity.

a
- + is a linear operator, so associativity will hold on $\mathbb{N}$.
- 0 is importent under +, so unitivity will hold.

b
- right-concatenation is associative by definition: $\forall a, b, c : (a; b); c = ab; c = abc \wedge a; (b; c) = a; bc = abc$
- $\forall x : []; x = x \wedge x; [] = x$. Unitivity is satisfied

c The object doesn't matter. The only thing that matters is the (implicit) composition table for the morphisms. The only time you need multiple objects is if some morphisms can't be applied after others. If every morphism is permitted at any time, only a single object is needed. Again, the structure of a category is in the composition table; objects simplify our understanding of the composition table by showing when it is prohibited to compose two morphisms.

Question 6

a 1

b this is equivallent to asking: "if a is divisible by b, and b is divisble by c, is a divisible by c?" yes. If $a/b = k_1$ and $b/c = k_2$, then $a/c = k_1 k_2$. So composition of two morphisms is multiplication: $f; g = f * g$

c If we added zero then you have to decide how to handle the Hom-set construction operator.

- $\forall y > 0 : P(0, y) = \emptyset$
- $\forall y > 0 : p(y, 0) = 0$
- $P(0, 0) = x : 0 * x = 0$ which is an infinite set, which violates the preorder definition that there only be a single morphism between any two objecst.

So, no. given the Hom-set generator, Zero violates the preorder condition. $a \cdot n$

Question 7

1. (AND $t)f$

$$((\lambda p.(\lambda q.(pq)p))t)f$$
$$((\lambda q.(tq))t)f$$
$$(tf)t$$
$$((\lambda x.(\lambda y.x))f)t$$
$$(\lambda y.f)t$$
$$f$$

2. (OR $f)t$

$$((\lambda p.(\lambda q.(pp)q))f)t$$
$$(\lambda q.(ff)q)t$$
$$(ff)t$$
$$((\lambda x.(\lambda y.y))f)t$$
$$(\lambda y.y)t$$
$$t$$

Question 8

$$Yg$$
$$(\lambda f.((\lambda x.f(xx))(\lambda x.f(xx))))g$$
$$(\lambda x.g(xx))(\lambda x.g(xx))$$
$$g((\lambda x.g(xx))(\lambda x.g(xx)))$$
$$g(g((\lambda x.g(xx))(\lambda x.g(xx))))$$
$$...$$

The pattern repeats forever: $g(g(...g((\lambda x.g(xx))(\lambda x.g(xx)))...))$

Question 9

See accompanying code.

Question 10

Question 9 was not well worded. Otherwise A+