Ryan Pollack

Sets

1. a. $\bar{A}$
   b. $A - B$
   c. $A \oplus B$

2. Let $P(x)$ be the power set of the set x, where its elements are the subsets of x.
   Suppose $A \subseteq B$
   If S represents an element and $S \in P(A)$ means that S is a subset of A, then $S \subseteq A \subseteq B$ shows that S is also a subset of B. Thus $S \in P(B)$
   Since every element of $P(A)$ is also an element of $P(B)$, we can conclude that $P(A) \subseteq P(B)$.

3. Let $n \in \mathbb{Z}$ be an integer
   Let $\mathbb{O}$ be the set of odd numbers and $\mathbb{E}$ the set of even numbers.
   By Division Theorem,
   $\forall n \in \mathbb{Z} : \exists : q, r \in \mathbb{Z}$ so that
   $n = 2q + r, \quad 0 \leq r < 2$
   which follows that
   $n = 2q \in \mathbb{E}$ or $n = 2q + 1 \in \mathbb{O}$
   Thus:
   each element of $\mathbb{Z}$ is no more than one of $\mathbb{E}$ or $\mathbb{O}$
   each element of $\mathbb{Z}$ is in at least one of $\mathbb{E}$ or $\mathbb{O}$.
   setting $q = 0$, then $0 \in \mathbb{E}$ and $1 \in \mathbb{O}$ so neither is empty
   Therefore, $\{\mathbb{E} \mid \mathbb{O}\}$ is a partition of $\mathbb{Z}$ by definition.

4. $(A - B) - C$
$= (A - B) \wedge \bar{C}$ $(A - B = A \wedge \bar{B})$
$= (A \wedge \bar{B}) \wedge \bar{C}$
$= A \wedge \bar{B} \wedge \bar{C}$
$= A \wedge \bar{C} \wedge \bar{B}$ (associative law)
$= (A \wedge \bar{C}) \wedge \bar{B}$
$= (A \wedge \bar{C}) - B$ $(A \wedge \bar{B} = A - B)$
$= (A - C) - B$
Therefore, $(A - B) - C = (A - C) - B$

Functions

1. Let S_N be the set of numbers $\{1, 2, 3, \ldots, N\}$
$|S_N| = N$
If $|x| = m$ and $|y| = n$ (variables are positive integers)
Then there are n^m functions $f: x \to y$.
Therefore, the answer for both is N^N.

2. A function is surjective if each element of the codomain is covered by at least one element of the domain. A function is bijective when it associates each element of the codomain with a unique element of the domain. Not all surjective functions are bijective, but all bijective functions are surjective. $S_N \to S_N$ when 1-to-1 cover both.

3. A function is injective if each element of the codomain has at most one element of the domain that associates with it. Bijective, again, is unique 1-to-1 matching. So, every bijective function is injective, but not every injective is bijective. $S_N \to S_N$ when 1-to-1 cover both.
surjective = ≥ 1 element each +
injective = ≤ 1 element each
Bijective is both surjective and injective at 1 element each.

4. N!

Cardinality

1. $|\bar{A} \cap \bar{B}|$

$= |\overline{A \cup B}|$

We know that for any set S, $|\bar{S}| = |U| - |S|$. So,

$= |U| - |A \cup B|$

$= |U| - (|A| + |B| - |A \cap B|)$

$= |U| - |A| - |B| + |A \cap B|$

2. Consider that f is a function from A to B and A, B are finite sets.. The cardinality of a set S, $|S|$ is the number in the set when it's finite.

Let's say $|S| = n$, where n is the elements contained in set S.

Then set $f(S)$ is the range of the set S, thus $f(S)$ contains all of the elements in the image of S.

So,

$f(S) = \{f(x) \mid x \in S\}$

Since S contains n elements, there are at most n elements of $f(x)$ with $x \in S$.

Thus,

$|f(S)| = |\{f(x) \mid x \in S\}| \leq n = |S|$.

Therefore,

$|f(S)| \leq |S|$.

3. Knowing that reals are uncountable and that rationals are countable, we know irrationals are uncountable because they are reals (uncountable) with the rationals (countable) removed.

Computability

1. For computer programming, we know that there are a finite amount of characters in the language and every program is finite. Then the set of all programs is countable since it's a subset of all the countable finite strings in the language.

2. To generate an output, a program must be finite, it must contain a finite number of characters with respective programming language, and it should compile without errors. Real numbers are uncountable so it's not standard.

3. No, two programming languages can be combined, however each language and computer memory is limited. There are an infinite amount of real numbers so you can never get unlimited numbers in a limited range.

4. No. Every program is a finite length string of characters over a finite alphabet, leading to a finite result. So, the result would likely be a very large number, but it should be countable. What the programs work with are already finite so them being uncountable infinite still bounds what they can do, so it shouldn't make a difference.

5. We are arguing that there are an uncountably infinite number of functions, or f: N → N. The number of functions from a set of size A to a set of size B is A·B. For infinite sets like N, this gives N·N, which would be uncountably infinite multiplied by itself, staying uncountably infinite. There is no way to enumerate all the functions from N to N, Therefore, there are functions from natural numbers to natural numbers (N → N) that are uncountable by any computer program.