CS Bridge Practice Exam 1

Note that the questions in this practice exam are not reflective of the true difficulty on the upcoming exam. Some questions will be harder than the ones you will see on the exam while other questions will be easier. Take this practice exam under exam conditions after you feel that you have a good grasp of all the material that will be tested.

1. Suppose that $A = \{1, 2, 3, \{1, 2, 3\}, \{2, 3\}\}$, determine if the following is true or false:

   (a) $\{2, 3\} \in A$

   This is true. Inside A, there is the element $\{2, 3\}$

   (b) $\{2, 3\} \subset A$

   This is true. Inside A, you can take the element 2 and 3 to form the subset $\{2, 3\}$

   (c) $\{\{1, 2, 3\}\} \in A$

   This is false.

   (d) $\{\{1, 2, 3\}\} \subset A$

   This is true. You can create a set with the element $\{1, 2, 3\}$.

2. Convert the following to

   (a) Decimal to Two's Complement

   I messed up the previous question since you all only covered 8-bit Two's Complement.

   Here's a replacement question with the answer:

$$-43_{10} = (11010101)_{\text{Two's Complement}}$$

   First step would be finding the binary representation for 43.

$$43/2 = 21r1$$
$$21/2 = 10r1$$
$$10/2 = 5r0$$
$$5/2 = 2r1$$
$$2/2 = 1r0$$
$$1/2 = 0r1$$

   Collecting all of the remainders gives us 00101011.

   We then perform $1000000 - 00101011 = 11010101$

   (b) Binary to Hexadecimal

$$110011100_2 = (19C)_{16}$$

   Split it up into the following: $1|1001|1100 \rightarrow 1|9|C$

3. Find a function whose domain is the set of all integers and whose target is the set of all positive integers that satisfies each set of properties.

   (a) Neither one-to-one, nor onto.

   One possible solution: $f(x) = 1$

   (b) One-to-one, but not onto.

   One possible solution:

$$f(x) = \begin{cases} 3x & x > 0 \\ 3|x| + 1 & x \leq 0 \end{cases}$$

   (c) Onto, but not one-to-one.

   One possible solution: $f(x) = |x| + 1$

   (d) One-to-one and onto.

$$f(x) = \begin{cases} 2x + 1 & x \geq 0 \\ -2x & x < 0 \end{cases}$$

4. Proofs

(a) Proof by Contradiction: Suppose that $a, b \in \mathbb{R}$. If a is rational and ab is irrational, then b is irrational.

Suppose for the sake of contradiction that a is rational and ab is irrational and b is not irrational. Thus we have a and b rational, and ab irrational. Since a and b are rational, we know there are integers c,d, e, f for which $a = c/e$ and $b = e/f$. Then $ab = ce/df$, and since both ce and d f are integers, it follows that ab is rational. But this is a contradiction because we started out with ab irrational.

(b) Proof by Contrapositive: For any $k \in \mathbb{Z}$, if $3k + 1$ is even, then k is odd.

By the contrapositive, we have that if k is even, then $3k + 1$ is odd.

Suppose that $k = 2m$ for some $m \in \mathbb{Z}$. Then, we see that $3(2m)+1 = 2(3m) + 1$. We then have that $3m$ is some integer such that we have that $3k + 1$ is odd.

(c) Direct Proof: If $x, y \in \mathbb{Z}$ and are both odd, then $x + y$ is even.

$x = 2n+1$ and $y = 2m+1$ for $n, m \in \mathbb{Z}$. Then $x+y = 2n+2m+2 = 2(n + m + 1)$. Therefore, we have that $x + y$ is even.

5. Find the sum of each individual digit within an integer. You should prompt the user for an integer n. You can assume that the user will not input negative integers or doubles.

   For example:

   Given $n = 567$, the code should output: 18

   Given $n = 18$, the code should output: 9

```cpp
#include <iostream>

int main() {
  int n, sum = 0;

  std::cout << "Enter a positive integer: ";
  std::cin >> n;

  while (n != 0) {
    sum += n % 10;
    n /= 10;
  }

  std::cout << sum << std::endl;

  return 0;
}
```

6. Print out a diamond. Prompt the user for a number n which will serve as the maximum possible number of stars in a row.

```cpp
#include <iostream>

int main() {
  int n;

  std::cout << "Enter n: ";
  std::cin >> n;

  int initialSpaces = n-1;
  int stars = n - initialSpaces;

  for (int i = 0; i < 2 * n + 1; i++) {
    for (int j = 0; j <= initialSpaces; j++) {
      std::cout << " ";
    }
    for (int j = 1; j < stars; j++) {
      std::cout << "* ";
    }
    std::cout << std::endl;

    if (i < n) { initialSpaces--; stars++; }
    else { initialSpaces++; stars--; }
  }

  return 0;
}
```