CSE 351 Section 3 – Numerical Representation Limits

Exercise 1: Averaging

```
/* Returns the average of the numbers in arr */

float average(int* arr, int n) {
  int sum = 0;
  for (int i = 0; i < n; i++) {
    sum += arr[i];
  }
  return sum / n;
}
```

Issue? Sum and n are integers, so `sum/n` will be rounded down to an integer value before it is cast to a `float`.

Fix? Cast sum to a `float` before dividing, or declare it as a `float` instead of `int`.

Exercise 2: Bank Account

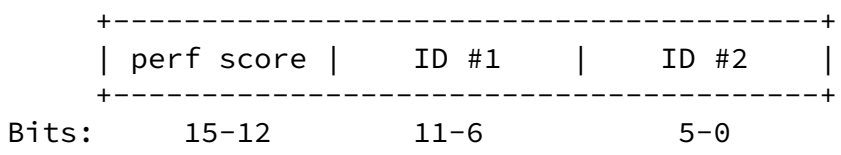
```
/* Subtracts the given amount from the bank account balance, then returns
the amount withdrawn */

float withdraw(float* balance, float amt) {
  if (amt > *balance) { return 0; }
  *balance -= amt;
  return amt;
}
```

Issue? If `balance` is significantly larger than `amt`, then `balance-amt` will round back up to `balance`, so the user can withdraw money without it actually being taken out of the account.

Fix? Check that `balance-amt != balance`.

Lab 1b Prep: Tennis! (Past Midterm Question)

Doubles tennis is played with 2 players to a side. Your tennis club is tracking the performances of player pairings using a 16-bit encoding scheme:



- Each player at the club has a unique ID.
- **Player IDs** in a pairing encoding must be distinct unsigned integers (*e.g.*, ID 0b111000 = ID 56).
- The **performance score** is encoded using Two's Complement, with more positive scores indicating better performance.

Exercise 3: Encodings

How many unique player IDs can be represented in this scheme?

ID fields are 6 bits wide, and each bit has 2 options (0/1) → 2^6 = **64** unique player ID possibilities.

Exercise 4: Limitations

What are the constraints/limitations on the range of values for the performance score?

The performance score is encoded within w = 4 bits, so we're limited to 2^4 = 16 possibilities. Since the score is encoded using Two's Complement, the most negative value we can represent is 0b1000 = -2^{w-1} = -2^3 = **-8 = TMin**, and the most positive number we can represent is 0b0111 = $2^{w-1}-1$ = 2^3-1 = **7 = TMax**.

Exercise 5: Bit Masking

Write a bit mask that will result in the binary of the performance score of a given pairing `x`. Your answer should have the 4 most significant bits set to the score and the 12 least significant bits set to `0`.

`x & 0xF000`

We want to "zero out" all bits except the first 4.

Exercise 6: Bit Manipulation

Write a series of bit manipulations (*i.e.*, shifting, using bitwise operators) to extract the ID of player #1 from a given pairing `x`. Your answer should have the 10 most significant bits set to 0 and the 6 least significant bits set to the ID of player #1.

`(x & 0x0FC0) >> 6`

First, utilize a bit mask to "zero out" all bits except the player #1's ID. Then, we shift the length of player #2's ID to move the leftover bits to the least significant end.