# CSE 351: Week 3

Tom Bergan, TA

# Today

- Questions on Lab 1 or Hw 1?
- Floating point
- Lab 2 quickstart

# The most important facts about floating-point numbers

- They are approximate
- Smaller numbers are more precise
  - think significant digits
  - I'll show you want I mean ....

# Floating point

## When you run this code

```
float x = 1.3;
printf(“%f\n”, x);
printf(“%.15\f”, x);
```

## It prints

```
1.300000
1.299999952316284
```

# Floating point

## When you run this code

```
float accountBalance = 1.30;
printf("%f\n", x);
printf("%.15\f", x);
```

**probably not a good idea**

- instead, maybe use:
  "binary-coded decimal" or
  "densely packed decimal"

## It prints

```
1.300000
1.299999952316284
```

# Floating point

**This code computes 1.3*10, right?**

```
float x = 1.3;
for(int i=0; i < 9; ++9)
   x += 1.3;

if (x == 13.0)
   printf("same!\n");
else
   printf("different!: %.15f\n", x);
```

**Not exactly ... it prints:**

```
different!: 13.000000953674316
```

# Floating point

## Here's a big number

```
float x = (float)((uint64_t)1 << 63);
printf("%f\n", x);
printf("%.15f\n", x);
```

## We can represent x precisely! (it's a power of 2)

## The code above prints

```
9223372036854775808.000000
9223372036854775808.000000000000000
```

# Floating point

**Now let’s add a small number to a big number**

```
float x = (float)((uint64_t)1 << 63);
x += 0.25;
printf(“%.15f\n”, x);
```

**The 0.25 disappears:**

```
9223372036854775808.000000000000000
```

# Floating point

## Doubles are more precise than floats

```
float x = 0.1;             // 32-bit floating point
double z = 0.1;            // 64-bit floating point
printf(“%.30f\n”, x);
printf(“%.30f\n”, x);
```

## But still approximate ... the above code prints:

```
0.100000001490116119384765625000
0.100000000000000005551115123126
```

# Floating point

## Floating point inaccuracy is hard to reason about

- how much error does ‘+’ introduce?
  - this is a hard numerical analysis problem
- compilers make this problem even harder
  - changing (x*1.3 + y*1.3) to 1.3*(x + y) could produce a different result

## See the work of William Kahn for the gory details

www.cs.berkely.edu/~wkahan
(Turing award winner for defining IEEE floating point numbers)

# Today

- ~~Questions on Lab 1 or Hw 1?~~
- ~~Floating point~~
- Lab 2 quickstart

**Demo**