EECS 280
Discussion #7

Week of February 18

Outline

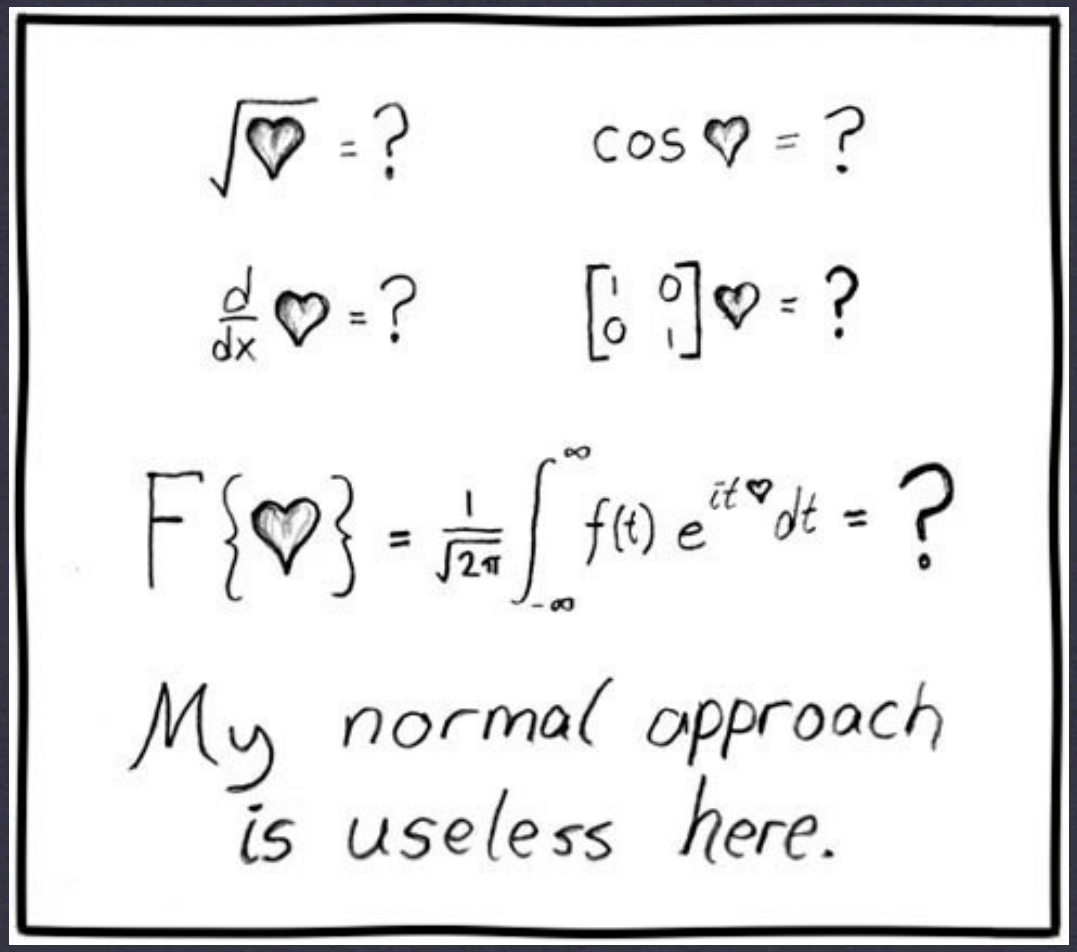
- **Administrivia**
- Exceptions
- Monopoly Testing

Administrivia

- Project 3
  - Due March 4 @ 11:59 PM
  - It's smooth sailing from here, right?
- Agenda: Post-Spring Break
  - Exam: Evening of March 6
  - Monday Class: Review
  - No Class on Wednesday
  - Discussion: Q&A, Sample Midterm

Outline

- Administrivia
- **Exceptions**
  - Motivation
  - Usage
  - Example
- Monopoly Testing



MOTIVATING HUMOR

THANKS XKCD

Motivating Example: Partay!

- Imagine you are throwing a pizza party and you want all invitees to be guaranteed the same number of slices of pizza
  - You know the number of guests (facebook/evite)
  - You know your pizza budget (i.e. number of pizzas)
  - You know how many slices/pizza

pizzaPerGuest

```
int pizzaPerGuest( int boxes, int guests )
{
   const int SLICE_PER_BOX = 8;

   return ( SLICE_PER_BOX * boxes ) / guests;
}
```

WHAT PROBLEMS COULD ARISE?

Better pizzaPerGuest

```
const int ERROR_DIV_BY_0 = -1;
const int ERROR_CODE2 = -2;
const int ERROR_CODE3 = -3;

int pizzaPerGuest( int boxes, int guests ) {
   const int SLICE_PER_BOX = 8;

   if ( !guests ) {
      cout << “eek!  divide by zero!” << endl;
      return ERROR_DIV_BY_0;
   }
   else if ( boxes < 0 )
   ...
   else
      return ( SLICE_PER_BOX * boxes ) / guests;
}
```

We can do this because of a limited function range!

Think back...

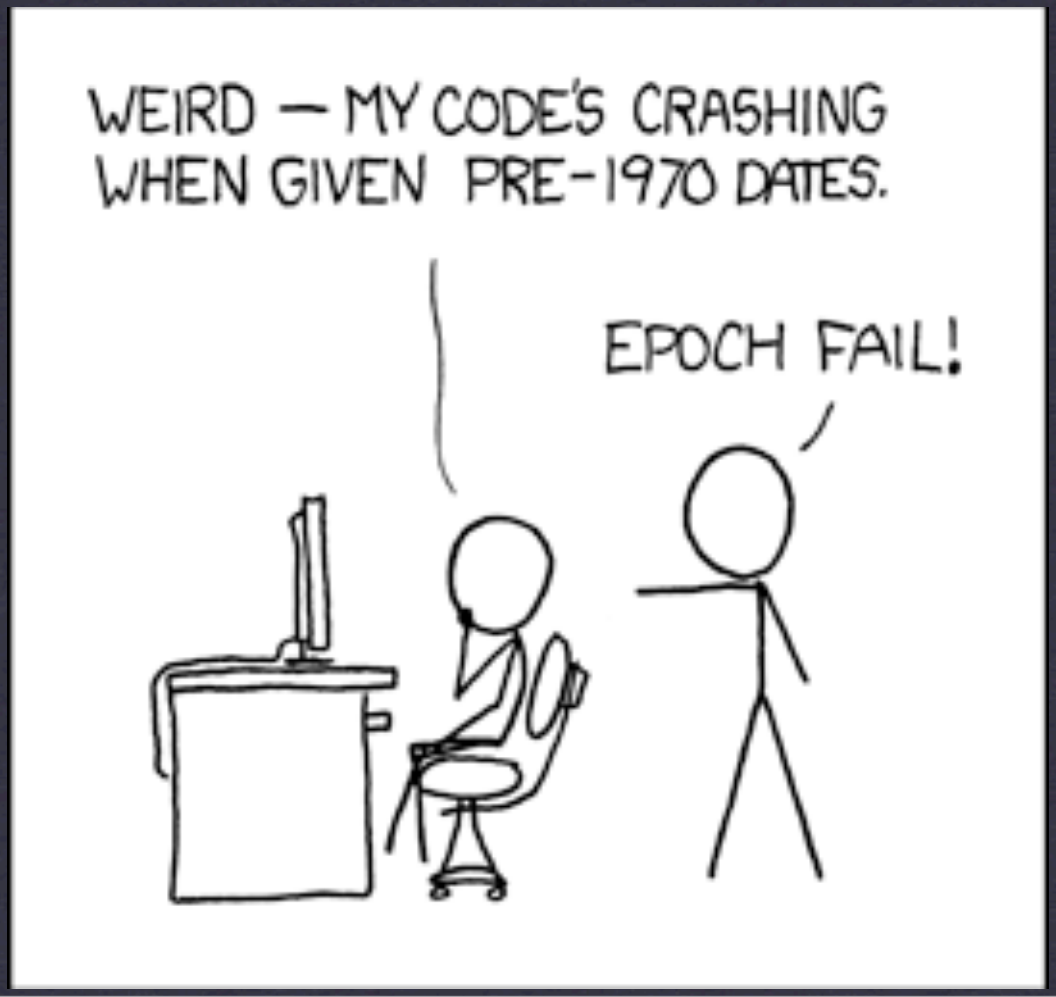
Slope of a line!

```
float get_slope(int x1, int y1, int x2, int y2)
// EFFECT: returns the slope of the line defined by
//         points (x1, y1) and (x2, y2)
{
   // rise over run!
   return ( ( y2 - y1 ) / ( x2 - x1 ) );
}
```

WHAT TO DO IN CASE OF A VERTICAL LINE?

Enter: The Exception

- Definition:
  - A condition that alters the flow of a program
- Basic Idea:
  - Indicate an expected, but unwanted, condition by waving a huge flag!
  - Hope that someone recognizes the flag...



EXCEPTION HANDLING

THANKS XKCD

Exception Usage

- Calling Function
  - Use try/catch blocks: try code, catch exceptions by type
- Called Function
  - Indicate types of expected exceptions in “throw list”
  - Throw desired types on respective exception condition

Example: Better Slope

```
double slope( int x1, int y1, int x2, int y2 ) throw ( char )
{
    if ( x1 == x2 )
        throw 'e';
    return ( ( y2 - y1 ) / ( x2 - x1 ) );
}
```

Example: Calling Slope

```
int main()
{
    try {
        cout << slope( 1, 1, 1, 2 );
    }
    catch ( char e ) {
        cout << "Vertical Line!";
    }
    return 0;
}
```

Example: Extensible Slope

```
struct my_error_type {
    int err_no;
    const char *err_msg;
};

double slope( int x1, int y1, int x2, int y2 ) throw ( my_error_type )
{
    if ( x1 == x2 ) {
        my_error_type e = { 1, "Vertical Line!" };
        throw e;
    }

    return ( ( y2 - y1 ) / ( x2 - x1 ) );
}
```

Example: Calling Slope (2)

```
int main()
{
    try {
        cout << slope( 1, 1, 1, 2 );
    }
    catch ( my_error_type e ) {
        cout << e.err_no << ": " << e.err_msg;
    }
    return 0;
}
```

Exceptions: Final Thoughts

- Functions can throw any [number of any] variable type, including user-defined data types
  - Consider: empty structs, error classes (and inheritance!!)
- A throw proceeds up through the stack until a suitable catch is found (else terminate)
- Nesting of exceptions is allowed
- Exceptions are a form of “goto” and should be avoided if possible

Outline

- Administrivia
- Exceptions
- **Monopoly Testing**
  - Dice Manipulation
  - Good Test Cases
  - Automating Testing
  - What is Concerning You?

Have a great break :)