# Introduction

Some simple ideas, and questions that I have about implementation and the language that I'm interested in designing.

# Idea (Come back to this lol)

conditionals as variables... using the keyword `cond` to indicate a conditional type

```
// this is bounding each naming to a reusable condition
cond {xGreaterThanYReturnX | yGreaterThanXReturnY | xAndYAreEqual} =
  | (x > y) => x  // binds x to this condition - and you can only use x
  | (x < y) => y
  | (x == y) => x;



/**
* Potentially better solution for the conditional expression
**/
const values = [1, 2, 3, 4, 5, 6 ,7];

cond {xGreaterThanYReturnX | yGreaterThanXReturnY | xAndYAreEqual} =
  | (x > y) => x & y;
  | (x < y) => y;
  | (x == y) => x;

// How to use the Condition Expression
xGreaterThanYReturnX(x = 5, y = 20) => {
  .
  <statement>
  .
}; // This will not execute x < y

yGreaterThanXReturnY(x = 5, y = 20) => {
  .
  <statement>
  .
}; // This will not execute x < y


// or if you have an expression already
const comparison x y => x = 10 ^ y = 20;

xGreaterThanYReturnX(comparison) => {
  .
  <statement>
  .
} else { // Will execute else condition in this case, 10 < 20
  .
  <statement>
  .
};
```

```
// feeling cool? Let's match these conditions:

const checkTwoNumbers x y = () => {
  match (...) { // (...) means all inputs being passed as arguments.
    | xGreaterThanYReturnX => x + 15; // whatever the value is of x + 15
    | yGreaterThanXReturnY => y + 15; // whatever the value is of y + 15
    | xAndYAreEqual => x + y + 15; // whatever the total value is + 15
  }
}
```

## Potential typing ideas

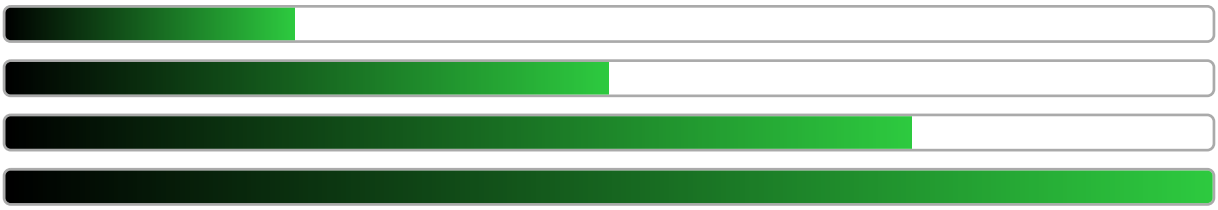
```
const checkTwoNumbers x y = (int, int) => {
  match (...) { // (...) means all inputs being passed as arguments.
    | xGreaterThanYReturnX => x + 15; // whatever the value is of x + 15
    | yGreaterThanXReturnY => y + 15; // whatever the value is of y + 15
    | xAndYAreEqual => x + y + 15; // whatever the total value is + 15
  }
}


// That way in the future if we do parametric polymorphism It might not be messy?

const checkTwoNumbers x y = (N, M) => {
  match (...) { // (...) means all inputs being passed as arguments.
    | xGreaterThanYReturnX => x + 15; // whatever the value is of x + 15
    | yGreaterThanXReturnY => y + 15; // whatever the value is of y + 15
    | xAndYAreEqual => x + y + 15; // whatever the total value is + 15
  }
}
```

=question

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ | $F_7$ | $F_8$ |



something

- [ ] Write code
- [ ] Test code
- [ ] Ship code