Anyone have any general questions?
A recursive algorithm is an algorithm which calls itself with "smaller (or simpler)" input values, and which obtains the result for the current input by applying simple operations to the returned value for the smaller (or simpler) input.

Implemented by having a method call itself with a simpler problem
basic parts of a recursion

- base case: This is the case we already know the answer to
- recursive part: call the current method with a smaller input
  - keep doing until you get to base case
recursion is just calling a method

A simple way to think of things is that we know how to do part of a problem but need someone else to do the rest.

So we do the part we can, and pass a smaller problem off to someone else.

That someone else just happens to do the same thing we do!
Recursive Example 1

//calculate the exponential value
int base=2;
int exponent=3;
int power(int base, int exponent)
{
    if(exponent<=0)
        return 1;
    return base*power(exponent-1);
}

wait! can't I do that in a loop?

//calculate the exponential value
int base=2; int exponent=3;
int power(int base, int exponent)
{
    int total=1;
    for(int i=exponent; exponent>0; exponent--)
        total*=base;
    return total
}
so why bother?

● the power() example was a simple one
● There are some algorithms that recursion is less code
  ○ adding nodes to a tree for example
    ■ more on what "trees" are in cse12
  ○ anything where you need to save current state for processing after recursive call
  ○ Some languages recursion is all you get (no loops)
  ○ For now trust us. It's useful

● Use the one that is simpler to write
  ○ make the computer work for you!
//add highest value in array to all other values
int [] val={5,1,9,4}
int max=Integer.MIN_VALUE; //smallest int ever
void updateArr(int[] val, int index)
{
    if(val[index]>max)
        max=val[index];
    updateArr(val, index);
    val[index]+=max;
}
//bad implementation but why?
fixed recursion pitfalls

```java
int [] val={5,1,9,4}
int max=Integer.MIN_VALUE; //smallest int ever
void updateArr(int[] val, int index)
{
    if(index>=val.length) //needed base case
        return;
    if(val[index]>max)
        max=val[index];
    updateArr(val, index+1); //recurse on smaller problem
    val[index]+=max;
}
```
int [] val={5,1,9,4}
int max=Integer.MIN_VALUE; //smallest int ever
void updateArr(int[] val, int index)
{
    if(index>=val.length) //needed base case
        return;
    if(val[index]>max)
        max=val[index];
    max=val[index];
    updateArr(val, index+1); //recurse on smaller problem
    val[index]+=max;
}
Exam Review Sheet

Keep in mind the review sheet is intended to be HARDER than the actual exam. If you are good with review, you will be good with the exam.

For now we'll look at a different review example (not on the review sheet)

TO THE CODE!
YOU KNOW THIS METAL
RECTANGLE FULL OF
LITTLE LIGHTS?

YEAH.

I SPEND MOST OF MY LIFE
PRESSING BUTTONS TO MAKE
THE PATTERN OF LIGHTS
CHANGE HOWEVER I WANT.

SOUNDS GOOD.

BUT TODAY, THE PATTERN
OF LIGHTS IS ALL WRONG!

OH GOD! TRY
PRESSING MORE
BUTTONS!

IT'S NOT
HELPING!