WEEK 7 STUDIO

MATRICES

Write a function `matrix` that takes in a list of lists that represents a matrix. This function returns a matrix "object" which gives the entry at a given row and column

```
const lst = list(list(1, 2, 3),
                 list(4, 5, 6),
                 list(7, 8, 9));

const m = matrix(lst);
// m(1)(1) === 1;
// m(2)(2) === 5;
// m(3)(1) === 7;
```

MATRICES

```
function matrix(seq) {
    return r => c => list_ref(list_ref(seq, r - 1), c - 1);
}
```

FIND SUM

Given a list of numbers and a number, write a function `find_sum` that returns a list of lists containing all possible combinations of numbers that add up to the sum.

```
find_sum(list(1, 2, 3, 4, 5), 5);
// returns [[5, null], [[2, [3, null]], [[1, [4, null]], null]]]
```

FIND SUM

```
function find_sum(lst, sum) {
    if (sum === 0) {
        return list(null);
    } else if (is_null(lst) || sum < 0) {
        return null;
    } else {
        // don't use the head
        const no_use = find_sum(tail(lst), sum);
        // uses the head
        const first = head(lst);
        const sub = find_sum(tail(lst), sum - first);
        const use = map(xs => pair(first, xs), sub);
        return append(no_use, use);
    }
}
```

CAN PARTITION?

Given a list of numbers and a number, write a function `can_partition` that returns `true` if the list can be partitioned into 2 portions of equal sum and `false` otherwise.

```
can_partition(list(1, 2, 3, 4)); // returns true
can_partition(list(1, 2, 4, 9)); // returns false
```