```
function mutable_reverse(xs) {
    function helper(curr, remain) {
        if (is_null(remain)) {
            return curr;
        } else {
            const push = tail(remain);
            set_tail(remain, curr);
            return helper(remain, push);
        }
    }
    return helper(null, xs);
}

function npr(s, r) {
    if (r === 0) {
        return list(null);
    } else if (is_null(s)) {
        return list(null);
    } else {
        const perm_per_elem = x => map(p => pair(x, p), npr(remove(x, s), r - 1));
        return accumulate(append, null, map(perm_per_elem, s));
    }
}

function append_iter(xs, ys) {
    function app(current_xs, ys, c) {
        return is_null(current_xs)
            ? c(ys)
            : app(tail(current_xs), ys, x => c(pair(head(current_xs), x)));
    }
    return app(xs, ys, x => x);

function similar(tn1, tn2) {
    if (is_null(tn1) && is_null(tn2)) {
        return true;
    } else if (is_number(tn1) && is_number(tn2)) {
        return math_abs(tn1 - tn2) <= 1;
    } else if (is_pair(tn1) && is_pair(tn2)) {
        return similar(head(tn1), head(tn2)) && similar(tail(tn1), tail(tn2));
    } else {
        return false;
    }
}

function memoize(f) {
    const mem = [];
    function mf(x) {
        if (mem[x] !== undefined) {
            return mem[x];
        } else {
```

```
            const result = f(x);
            mem[x] = result;
            return result;
        }
    }
    return mf;
}

function take(xs, n) {
    return n === 0
        ? null
        : pair(head(xs), take(tail(xs), n-1));
}

function drop(xs, n) {
    return n === 0
        ? xs
        : drop(tail(xs), n-1);
}

function member_list(xs, L) {
    if (length(L) < length(xs)) {
        return null;
    } else {
        if (head(xs) === head(L)) {
            const ys = take(L, length(xs));
            return equal(xs, ys)
                ? L
                : member_list(xs, tail(L));
        } else {
            return member_list(xs, tail(L));
        }
    }
}
```