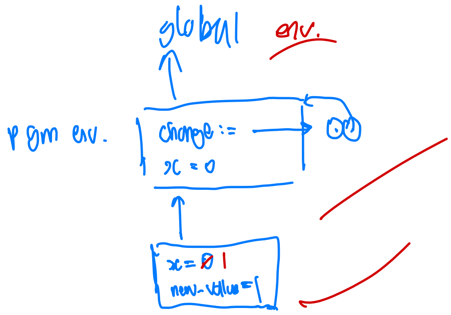
National University of Singapore
School of Computing
CS1101S: Programming Methodology
Semester I, 2021/2022

S9
Mutable Data and Environment Model

Problems:



1. Consider the following function declaration:

```
function change(x, new_value) {
    x = new_value;
}
```

Consider the following statements:

```
let x = 0;
change(x, 1);
```

What is the value of x after the above statements are evaluated? Why? Explain your answer using the environment model.

0 .

2. Write the function d_filter that takes as arguments a one-argument predicate function pred and a list xs, and returns a list that contains only those elements for which pred returns true. Your function must not create any new pair, and the result list must only be made of existing pairs in xs. Your function must not modify the head of any of the existing pairs.

can change the tail

```
function d_filter(pred, xs) {
    // ???
}
```

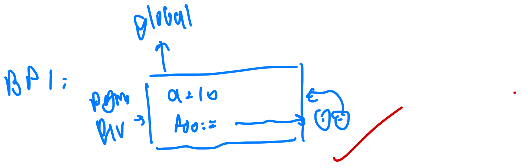
Example call:

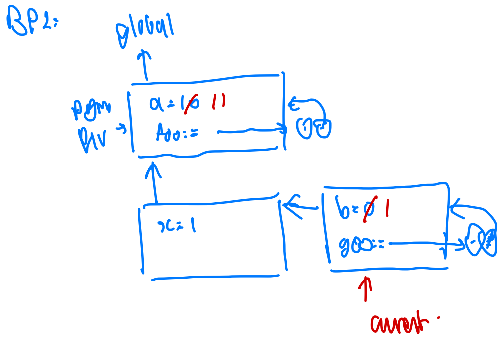
```
const L = list(1, 2, 3, 4, 5, 6, 7, 8, 9, 11);
d_filter(x => x % 2 === 0, L);  // returns [2, [4, [6, [8, null]]]]
L;  // What is L now? = list(1,2,4,6,8).
```

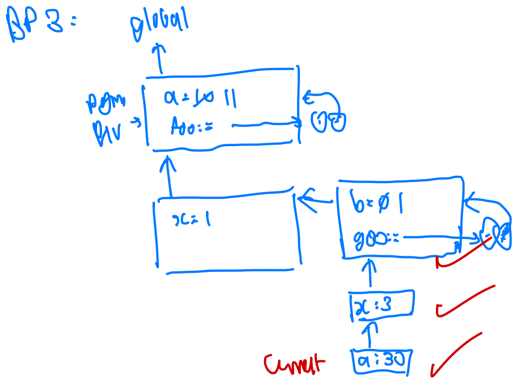
```
function d_filter(pred, xs){
    if (is_null(xs)){
        return null;
    }
    else if (pred(head(xs))){
        set_tail(xs, d_filter(pred, tail(xs)));
        return xs;
    }
    else {
        return d_filter(pred, tail(xs));
    }
}
const L = list(1, 2, 3, 4, 5, 6, 7, 8, 9, 11);
d_filter(x => x % 2 === 0, L);
```

3. Given the following Source program:

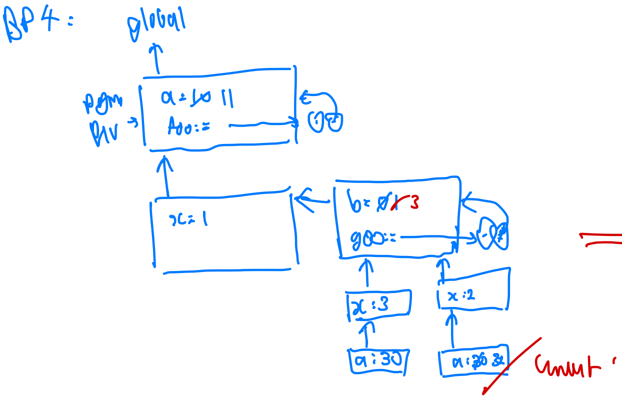
```
let a = 10;

function foo(x) {
    let b = 0;

    function goo(x) {
        let a = 30;

        if (x <= 2) {
            a = a + x;
            b = b + x;
            // Breakpoint #4
        } else {
            // Breakpoint #3
            goo(x - 1);
        }
    }

    a = a + x;
    b = b + x;
    // Breakpoint #2
    goo(3);
}

// Breakpoint #1
foo(1);
// Breakpoint #5
```

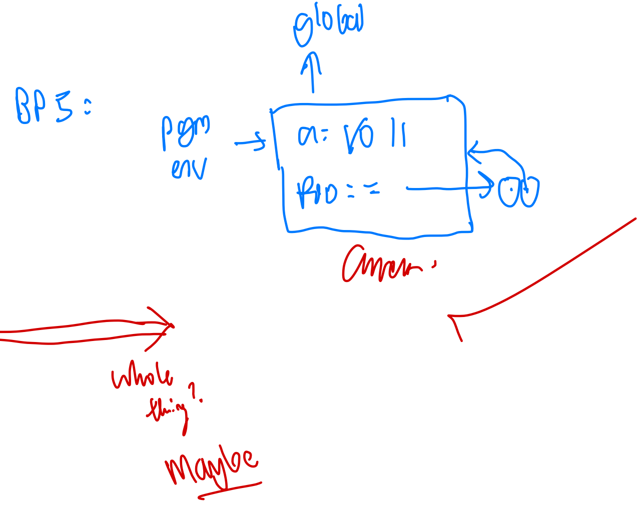






Evaluate the program and draw all the frames of the environment that have been created up to each breakpoint. Also indicate the **current environment** in the environment diagram at each breakpoint.





National University of Singapore
School of Computing
CS1101S: Programming Methodology
Semester I, 2021/2022

S9-in-class
Mutable Data and Environment Model

Problems:

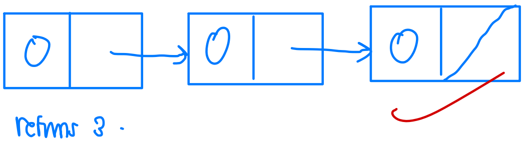
1. **[In-class only]** Ben Bitdiddle decides to write a function to count the number of pairs in any list structure. “It’s easy,” he reasons. “The number of pairs in any structure is the number in the head plus the number in the tail plus one more to count the current pair.” So Ben writes the following function:

```
function count_pairs(x) {
    if (!is_pair(x)) {
        return 0;
    } else {
        return 1 + count_pairs(head(x)) + count_pairs(tail(x));
    }
}
```

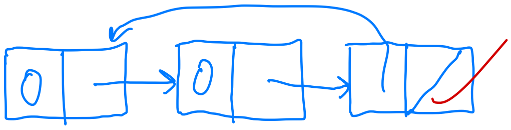
i) Show that this function is not correct. In particular, draw box-and-pointer diagrams representing list structures made up of exactly three pairs for which Ben’s procedure would return 3, return 4, return 7, or never return at all.

Devise a correct version of `count_pairs` that returns the number of distinct pairs in any structure. (Hint: Traverse the structure, maintaining an auxiliary data structure that is used to keep track of which pairs have already been counted.)

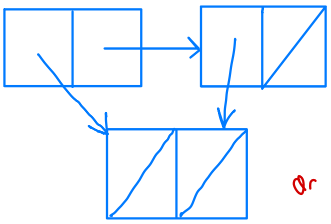
i)



returns 3 .

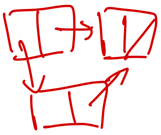


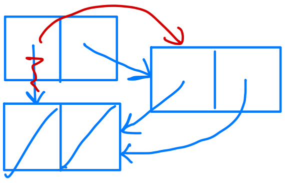
never return at all.



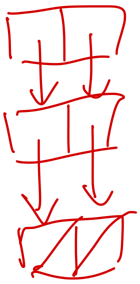
returns 4

or





returns 7



ii) function count_pairs(x) {
let length = 0;

→ store pointer inside an array.

→ once same pointer, return 0

→ else return 1. or use length

```
function count_pairs(x) {
    let pairs = null;
    function check(y) {
        if (!is_pair(y)) {
            return undefined;
        } else if (!is_null(member(y, pairs))) {
            return undefined;
        } else {
            pairs = pair(y, pairs);
            check(head(y));
            check(tail(y));
        }
    }
    check(x);
    return length(pairs);
}
```

if y is alr seen.

check current pair