NATIONAL UNIVERSITY OF SINGAPORE

CS1101S — PROGRAMMING METHODOLOGY

(Semester 1 AY2015/2016)

CURATED VERSION OF 16/11/2020 (LAST CORRECTED ON 8/12/2021)

Time Allowed: **2 Hours**

SOLUTIONS

INSTRUCTIONS TO STUDENTS

1. This assessment paper contains **SEVEN (7)** questions and comprises **TWENTY-TWO (22)** printed pages, including this page.
2. The full score of this paper is **80 marks**.
3. This is a **CLOSED BOOK** assessment, but you are allowed to use **TWO** double-sided A4 sheets of written or printed notes.
4. Answer **ALL** questions **within the space provided** in this booklet.
5. Where programs are required, write them in the **Source §4** language.
6. Write legibly with a **pen or pencil**. **UNTIDINESS will be penalized**.
7. Do not tear off any pages from this booklet.
8. Write your **Student Number** below **USING A PEN**. Do not write your name.

STUDENT NO.: _____

This portion is for examiner's use only

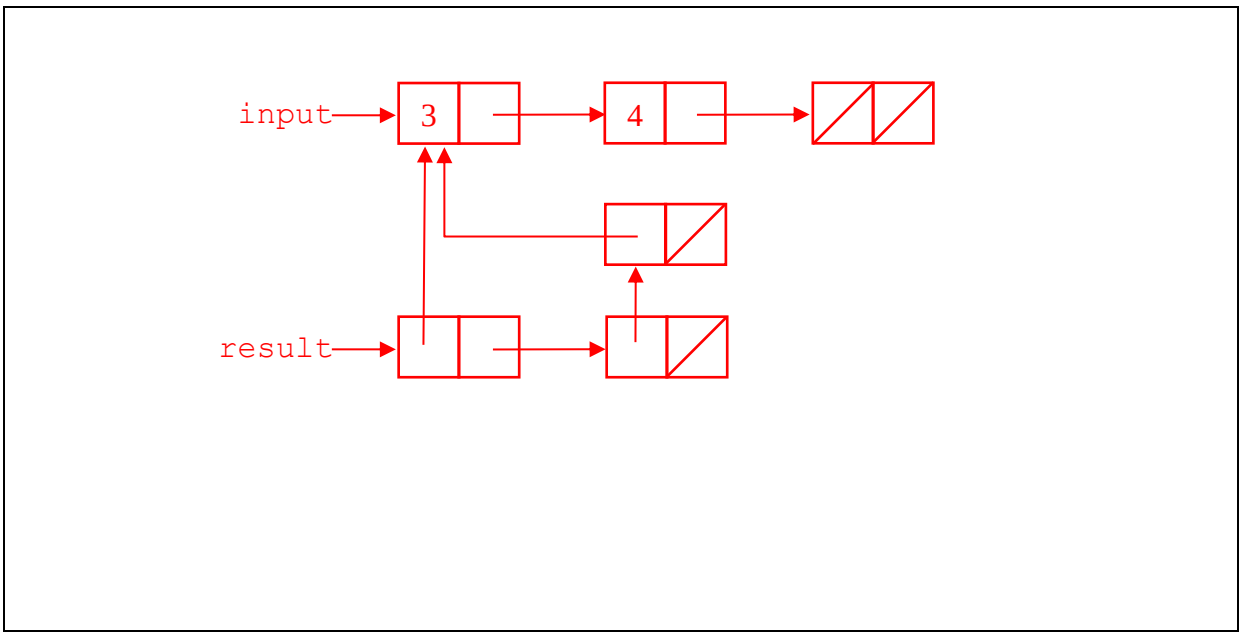
Question	Marks	Question	Marks
Q1 (14 marks)		**Q5 (8 marks)**	
Q2 (9 marks)		**Q6 (13 marks)**	
Q3 (12 marks)		**Q7 (17 marks)**	
Q4 (7 marks)		**TOTAL (80 marks)**	

Question 1: Boxes and Pointers [14 marks]

For each of the Source programs in Parts A, B and C, show the box-and-pointer diagram for `result` and `input` at the end of the execution of the program. Clearly show where `result` and `input` are pointing to.

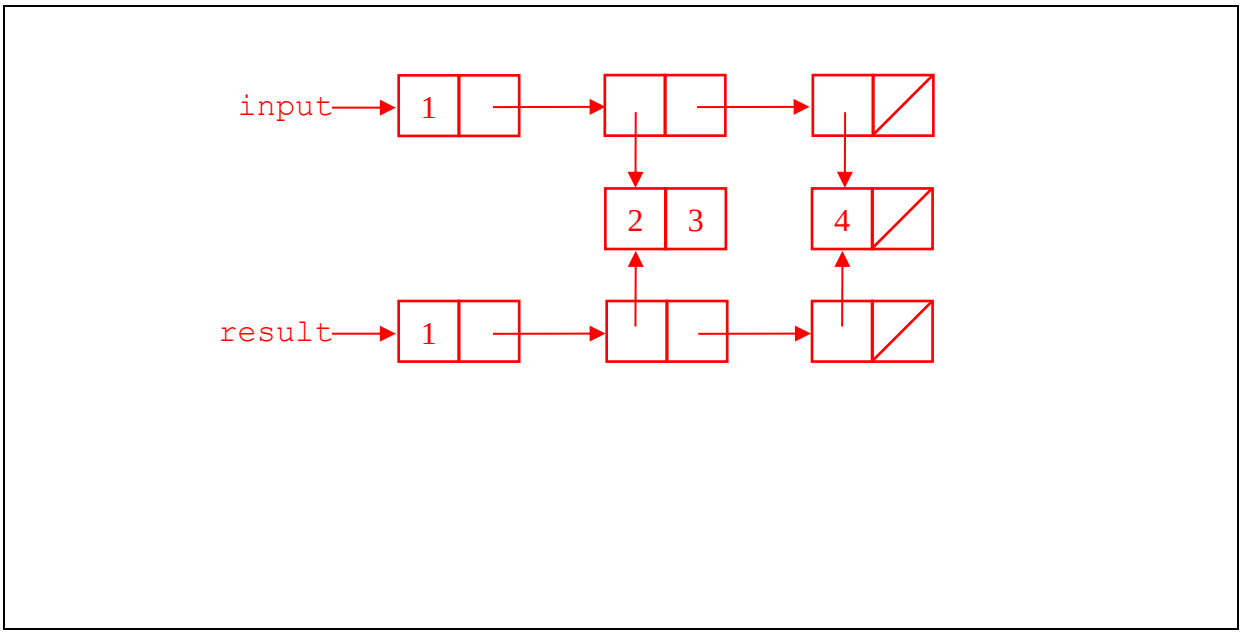
1A. [2 marks]

```
let input = list(3, 4, null);
let result = list(input, pair(input, null));
```



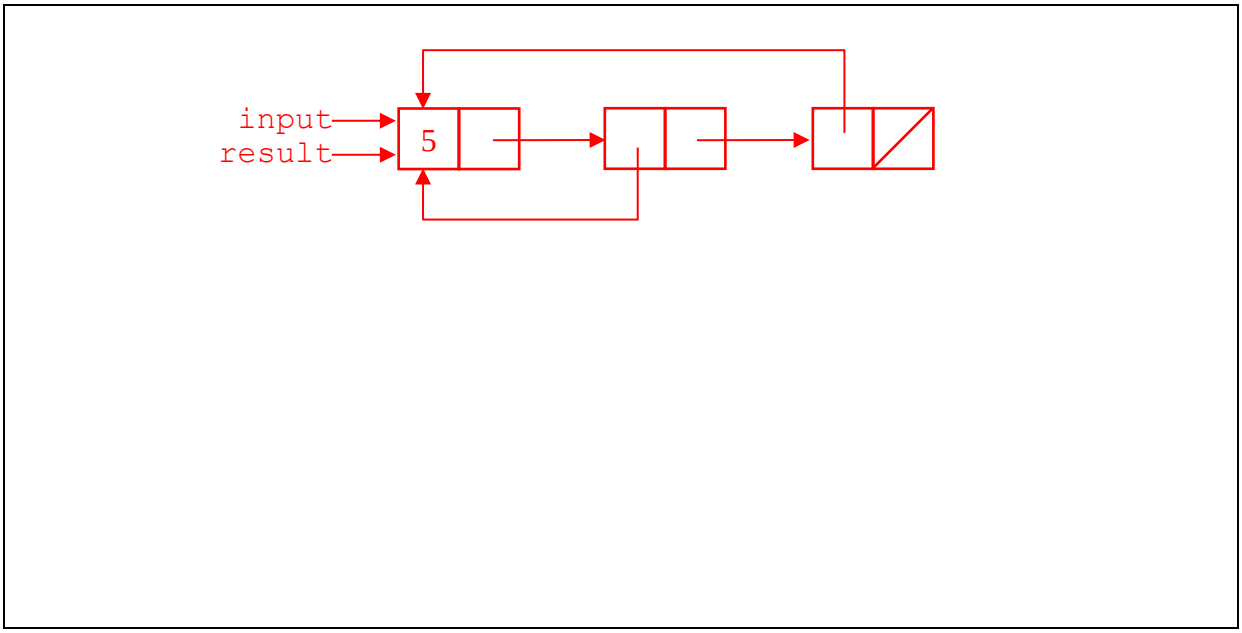
1B. [3 marks]

```
let input = list(1, pair(2,3), list(4));
let result = map(x => x, input);
```



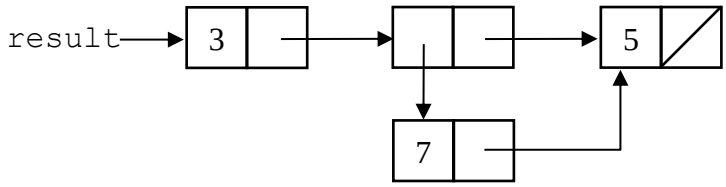
1C. [4 marks]

```
function m_map(f, xs) {
    if (is_null(xs)) {
        return xs;
    } else {
        set_head(xs, f(head(xs)));
        m_map(f, tail(xs));
        return xs;
    }
}
let input = list(1, pair(2,3), list(4));
let result = m_map(x => is_number(x) ? 5 * x : input, input);
```



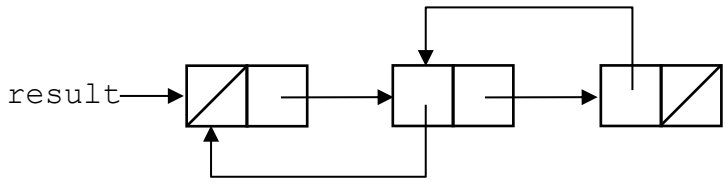
For each of Parts D and E, write a Source program that produces exactly the pairs shown in the box-and-pointer diagrams. At the end of the execution of your program, the identifier `result` must refer to the pair as shown in the diagram. You must not use `set_head` or `set_tail` unless it is absolutely necessary; otherwise marks will not be awarded for your solution.

1D. [2 marks]




```
let x = pair(5, null);
let y = pair(7, x);
let z = pair(y , x);
let result = pair(3, z);
```

1E. [3 marks]




```
let x = pair(null, null);
let y = pair(null, x);
let result = pair(null , y);
set_head(y, result);
set_head(x, y);
```

Question 2: The Environment Model [9 marks]

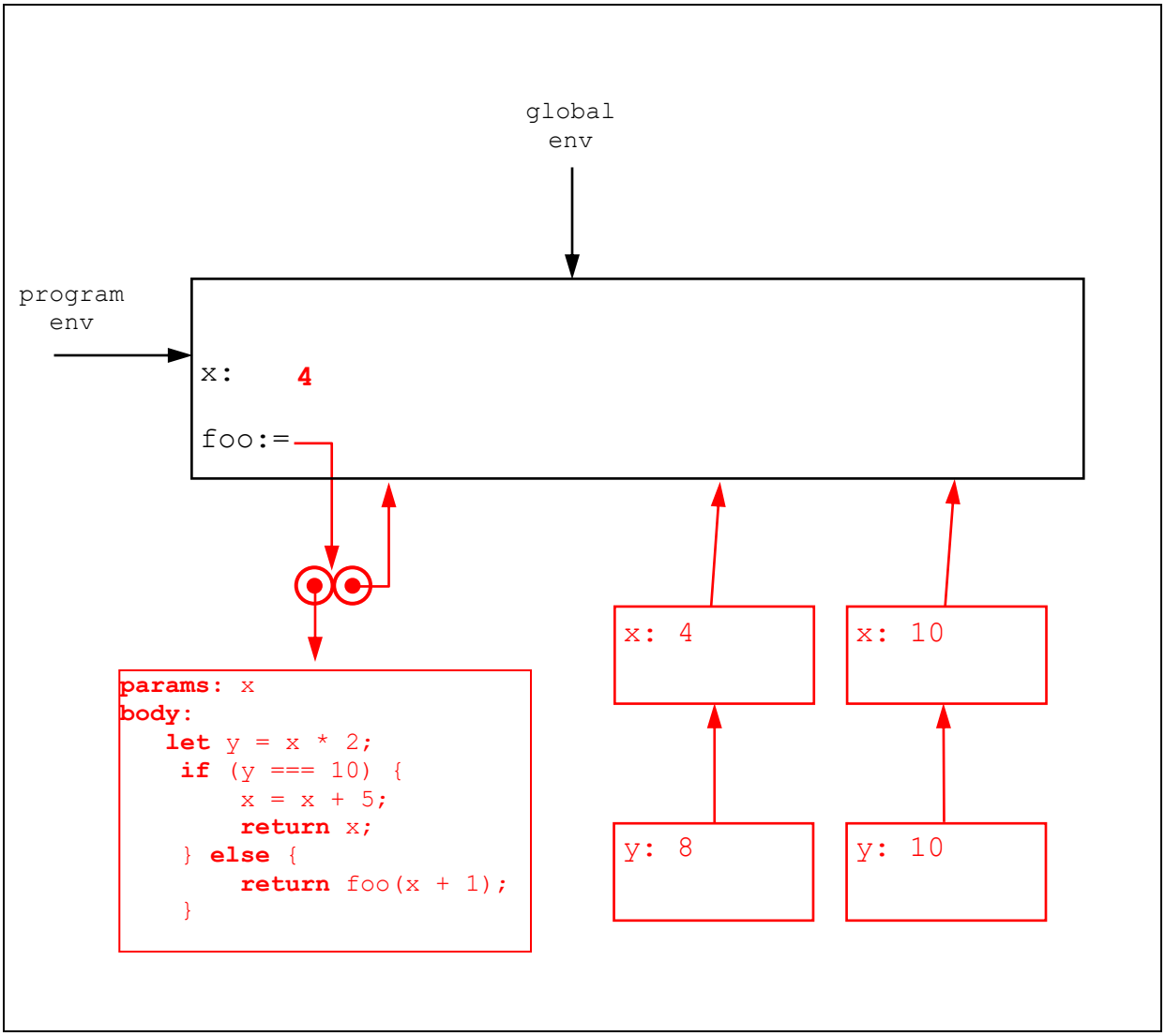
2A. [4 marks]

Given the following Source program, complete the following environment model diagram to show all the environments at the point of execution marked ***HERE***. Only draw non-empty frames.

```
let x = 4;

function foo(x) {
    let y = x * 2;
    if (y === 10) {
        x = x + 5;
        // HERE
        return x;
    } else {
        return foo(x + 1);
    }
}

foo(x);
```

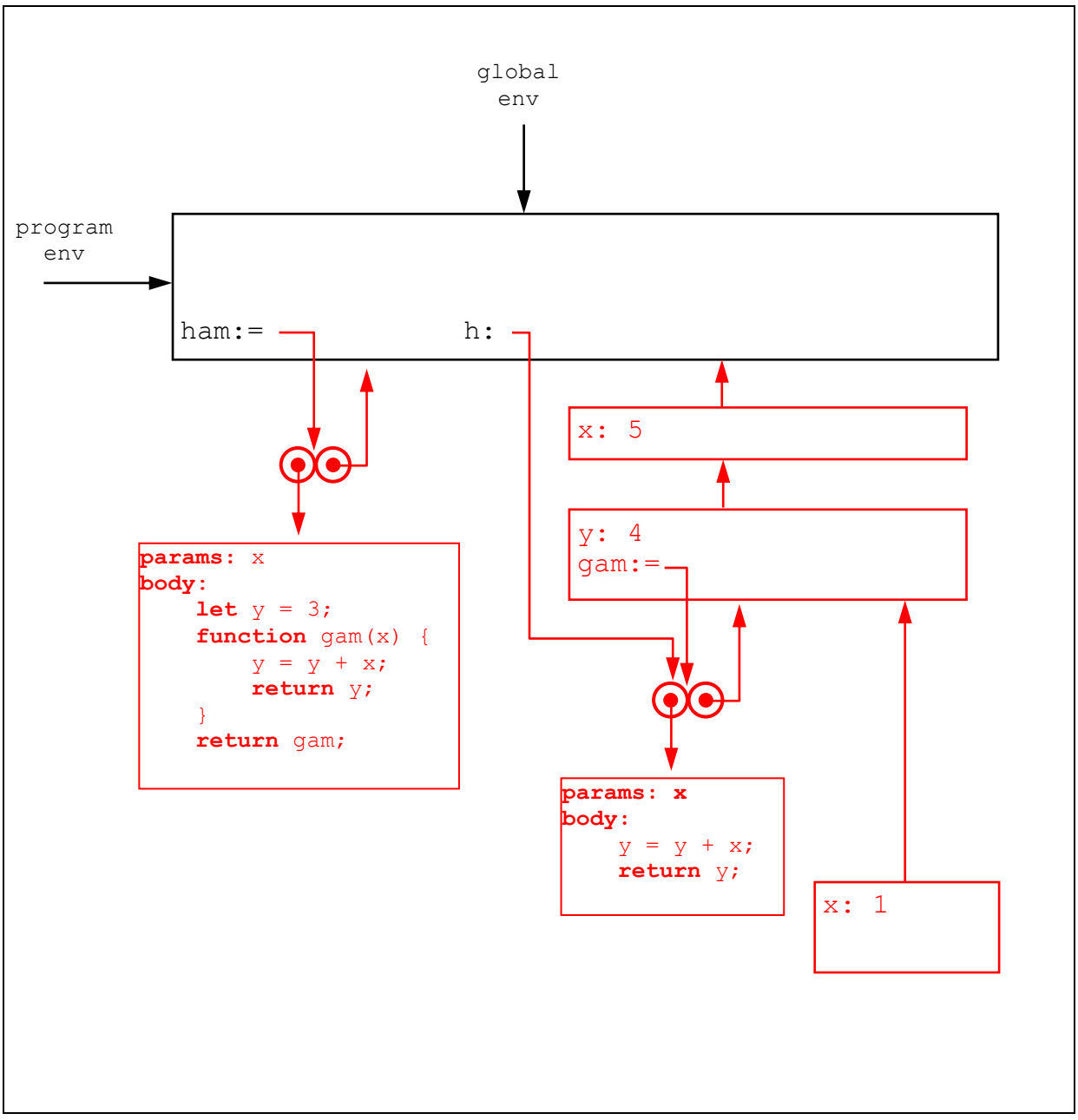


2B. [5 marks]

Given the following Source program, complete the following environment model diagram to show all the environments at the point of execution marked ***HERE***. Only draw non-empty frames.

```
function ham(x) {
    let y = 3;
    function gam(x) {
        y = y + x;
        // HERE
        return y;
    }
    return gam;
}

let h = ham(5);
h(1);
```



Question 3: Iterations and Recursions [12 marks]

We would like to use recursion to produce the same output as the following program that uses nested **for** loops:

```
let n = ...; // n is a positive integer number
for (let x = 1; x < n; x = x * 2) {
    for (let y = 0; y < x; y = y + 1) {
        display("x: " + stringify(x) + ", y: " + stringify(y));
    }
}
```

3A. [6 marks]

Complete the following program to use two recursive functions to produce the same output as the above nested **for** loops. You must not define any other function or use any loop.

```
let n = ...; // n is a positive integer number

function outer_loop(x) {

    function inner_loop(y) {

        if (y < x) {
            display("x: " + x + ", y: " + y);
            inner_loop(y + 1);
        } else {;}









    } // inner_loop

    if (x < n) {
        inner_loop(0);
        outer_loop(x * 2);
    } else {;}









} // outer_loop

outer_loop(1);
```

3B. [6 marks]

Complete the following program to use a single recursive function to produce the same output as the nested **`for`** loops. You must not define any other function or use any loop.

```
let n = ...; // n is a positive integer number

function double_loop(x, y) {

    if (x >= n) {
        ;
    } else if (y >= x) {
        double_loop(x * 2, 0);
    } else {
        display("x: " + x + ", y: " + y);
        double_loop(x, y + 1);
    }









}

double_loop(1, 0);
```

Question 4: Arrays [7 marks]

[7 marks]

Write a function `circular_right_shift(arr)` that takes a 2D rectangular array `arr`, and modifies the array by shifting all the elements to the right such that an element in the right-most column is shifted to the left-most column of the next row, and the bottom right-most element is shifted to the top left-most position.

For example, given that the 2D array, `arr`, of height 4 and width 3 is

```
let arr = [[ 1,  2,  3],
           [ 4,  5,  6],
           [ 7,  8,  9],
           [10, 11, 12]];
```

`circular_right_shift(arr)` modifies `arr` to become

```
          [[12,  1,  2],
           [ 3,  4,  5],
           [ 6,  7,  8],
           [ 9, 10, 11]];
```

You can assume the input array `arr` has at least 2 rows and at least 2 columns.

```
function circular_right_shift(arr) {
    let height = arr.length;
    let width  = arr[0].length;

    // VERSION 1
    let prev = arr[height-1][width-1];

    for (let r = 0; r < height; r = r + 1) {
        for (let c = 0; c < width; c = c + 1) {
            let temp = arr[r][c];
            arr[r][c] = prev;
            prev = temp;
        }
    }
```

(more writing space next page)

```
    // VERSION 2
    let last = arr[height-1][width-1];

    for (let r = height - 1; r >= 0; r = r - 1) {
        for (let c = width - 1; c >= 0; c = c - 1) {
            if (r === 0 && c === 0) {
                arr[r][c] = last;
            } else if (c === 0) {
                arr[r][c] = arr[r - 1][width - 1];
            } else {
                arr[r][c] = arr[r][c - 1];
            }
        }
    }



}
```

Question 5: Mutable List Processing [8 marks]

[8 marks]

Write a function `mutable_reverse(xs)` that takes in a list of numbers `xs`, and returns a list that is the reverse of the input list. Your function **must not create any new pairs**, and **every pair in the result list must be an existing pair of the input lists**. You must not modify the head of any pair.

For example:

```
let as = list(1, 2, 3, 4, 5);
let bs = mutable_reverse(as);
bs;  // equal to list(5, 4, 3, 2, 1).
as;  // equal to list(1).
```

```
function mutable_reverse(xs) {

    // VERSION 1
    if (is_null(xs)) {
        return xs;
    } else if (is_null(tail(xs))) {
        return xs;
    } else {
        let temp = mutable_reverse(tail(xs));
        set_tail(tail(xs), xs);
        set_tail(xs, null);
        return temp;
    }


    // VERSION 2
    function helper(prev, xs) {
        if (is_null(xs)) {
            return prev;
        } else {
            let rest = tail(xs);
            set_tail(xs, prev);
            return helper(xs, rest);
        }
    }
    return helper(null, xs);









}
```

The following question is not relevant for CS1101S as of 2019/20.

Question 6: The Game of Pig [13 marks]

The dice game *Pig* is a turn-taking dice game. The players start out with a score of 0 and the first player who reaches a score of 100 or more wins.

In addition to the current score, each player has a cache of points during his/her turn. At the beginning of a player's turn, the player's cache has 0 point. The player can decide to "hold" the cache or to "roll" the dice. If the decision is to "hold", the cache is added to the score and the next player continues. If the decision is to "roll", the player rolls the dice, the number on the dice is added to his/her cache, and the same player continues.

The catch: If a player rolls 1, his cache is forfeited (the score remains unchanged), and the next player continues.

Example: Alice and Bob start the game, each with a score of 0. Alice starts. She decides to "roll", and the dice shows 5. Now her score is still 0, but her cache is 5. She decides to "roll" again, and this time the dice shows 6. Now her score is still 0, but her cache is 11. She decides to "hold". Now her score is 11 and her cache is 0. Bob decides to "roll", and the dice shows 4. His score is 0, but his cache is 4. He decides to "hold", so his score is 4 and his cache is 0. Alice decides to "roll", and the dice shows 2. Her score is 11 and her cache is 2. She decides to "roll", but this time, the dice shows 1. Her score remains 11, her cache is 0, and it is Bob's turn. The game continues.

6A. [3 marks]

Recall the unary function `Math.floor`, which returns the largest integer smaller than or equal to the number given as argument, and the nullary function `Math.random`, which returns a "randomly" generated number between 0 (inclusive) and 1 (exclusive). Statistically, the results are approximately evenly distributed over this interval.

Design a class `Dice` whose constructor function does not take any argument, and it has a method `roll` with no argument, which returns a number between 1 (inclusive) and 6 (inclusive). Example:

```
let my_dice = new Dice();
display(my_dice.roll()); // displays 5
display(my_dice.roll()); // displays 2
display(my_dice.roll()); // displays 1
```

```
function Dice() {}

Dice.prototype.roll = function() {
    return Math.floor(Math.random() * 6) + 1;
};
```

6B. [2 marks]

In order to provide a fair Game of Pig, every player will use the same dice. So in addition to the player's name, the constructor function of the `Player` class will take a `Dice` object as argument. Remember that every `Player` of the game needs to keep track of his/her score and cache.

Write a function `Player` such that

```
let dice = new Dice();
let alice = new Player("Alice", dice);
let bob = new Player("Bob", dice);
```

sets up the game with players Alice and Bob.

```
function Player(n, d) {
    this.score = 0;
    this.cache = 0;
    this.name = n;
    this.dice = d;
}
```

6C. [2 marks]

The players take turns. Thus, we need to make sure they know of each other. One way to do this is to introduce a method `set_opponent` that allows a `Player` object to remember his/her opponent player. Add a method `set_opponent` to the `Player` class such that the following program sets up a Game of Pig, where Alice knows Bob as opponent and Bob knows Alice as opponent.

```
let dice = new Dice();
let alice = new Player("Alice", dice);
let bob = new Player("Bob", dice);
alice.set_opponent(bob);
bob.set_oppponent(alice);
```

```
Player.prototype.set_opponent = function(opp) {
    this.opponent = opp;
};
```

6D. [3 marks]

In the next part of this question (Part 6E), we will add a method `continu` to the `Player` class, which is called to let a `Player` object start his/her turn.

At the end of each turn (whether via "holding" or via "rolling" a 1), we need to check whether the player has won, in which case we need to announce the player as winner using his/her name (e.g. "Alice has won the game!"). If the player has not won, we need to set his cache to 0 and send the message `continu()` to the player's opponent.

Add the method `complete_turn` to the `Player` class.

```
Player.prototype.is_winner = function() {
    return this.score >= 100;
};

Player.prototype.continu = function() {
    // To be defined in Part 6E.
};

Player.prototype.complete_turn = function() {

    if (this.is_winner()) {
        display(this.name + " has won the game!");
    } else {
        this.cache = 0;
        this.opponent.continu();
    }

};
```

6E. [3 marks]

It is interesting to think about a strategy to play the Game of Pig. However, for the purpose of this assessment, we can simply ask the human player to decide whether to "hold" or "roll", as done by the following method `ai`.

```
Player.prototype.ai = function() {
    return prompt(this.name + ": hold or roll? ");
};
```

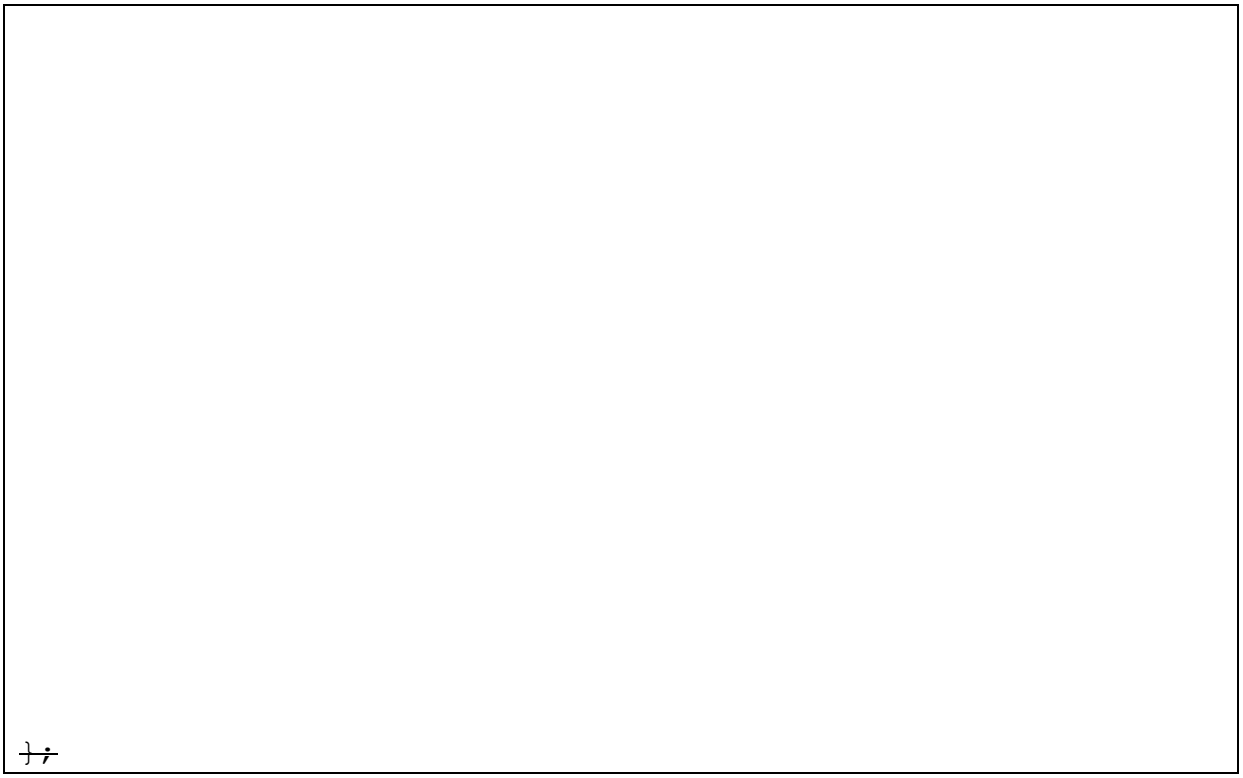
Add the method `continu` to the `Player` class such that the following program allows us to play a complete Game of Pig.

```
function pig() {
    let dice = new Dice();
    let alice = new Player("Alice", dice);
    let bob = new Player("Bob", dice);
    alice.set_opponent(bob);
    bob.set_opponent(alice);
    alice.continu();  // Alice starts first.
}
pig();
```

```
Player.prototype.continu = function() {

    let decision = this.ai();
    if (decision === "hold") {
        this.score = this.score + this.cache;
        this.complete_turn();
    } else { // (decision === "roll")
        let roll_result = this.dice.roll();
        if (roll_result === 1) {
            this.complete_turn();
        } else {
            this.cache = this.cache + roll_result;
            this.continu();
        }
    }
```

(more writing space next page)



Question 7: Treams [17 marks]

A *tream* is a stream whose elements are data items, or treams.

7A. [3 marks]

Construct a tream `t` that contains two elements, where the first element is a tream with the elements 2 and 4, and the second element is a tream with the elements 3 and 5. For example, with your constructed tream `t`, the following statements read the data items in `t`:

```
head(head(t));  // returns 2.
head(stream_tail(head(t)));  // returns 4.
head(head(stream_tail(t)));  // returns 3.
head(stream_tail(head(stream_tail(t))));  // returns 5.
```

```
let e = () => null;
let a = pair(2, () => pair(4, e));
let b = pair(3, () => pair(5, e));
let t = pair(a, () => pair(b, e));
```

7B. [5 marks]

A *binary tream* is a tream with three elements, where the first element is a data item, and the second and third elements are binary treams. The first element is called the tream's *node data*, the second element is called the tream's *left child*, and the third element is called the tream's *right child*.

Construct a binary tream `b` whose node data is the number 1. For each tream `b'` in `b`, the node data of the left child of `b'` should be 1 larger than the node data of `b'`, and the node data of the right child should be twice as large as the node data of `b'`.

```
function bin_tream(num) {
    let e = () => null;
    let z = () => pair(bin_tream(num * 2), e);
    let y = () => pair(bin_tream(num + 1), z);
    return pair(num, y);
}

let b = bin_tream(1);
```

7C. [5 marks]

Recall that a *tree* is a list whose elements are data items, or trees. Write a function `tree_to_tream` that converts a given tree into a tream.

```
function tree_to_tream(tree) {

    if (is_null(tree)) {
        return null;
    } else {
        let x = () => tree_to_tream(tail(tree));
        if (is_pair(head(tree))) {
            return pair(tree_to_tream(head(tree)), x);
        } else {
            return pair(head(tree), x);
        }
    }









}
```

7D. [4 marks]

Write a function `tream_map` that takes a function `f` and a tream `t` as arguments, and returns a tream that is just like `t`, except that every data item `d` is replaced by `f(d)`.

```
function tream_map(f, t) {

    if (is_null(t)) {
        return null;
    } else {
        let x = () => tream_map(f, stream_tail(t));
        if (is_pair(head(t))) {
            return pair(tream_map(f, head(t)), x);
        } else {
            return pair(f(head(t)), x);
        }
    }

}
```

(Scratch paper. Do not tear off.)

(Scratch paper. Do not tear off.)

——— END OF PAPER ———