Week 03 Studio
Substitution Model, Recursion

CS1101S AY21/22 Semester 1
Studio 05E

Yan Xiaozhi (David)
@david_eom
yan_xiaozhi@u.nus.edu

23 Aug 2021


ROPE
ASIA
MERI

Admin

- Contact tracing (QR code + class photo)
  - Remind me if I forget plezzzz 🙏
- Mission 2A (Rune Reading) marking and debrief
  - Unsubmission policy
- Mission 2B (Beyond the Second Dimension) due tmr night 2359!
  - Please do not rush mission last minute!
- Styling!!! For later mods / internship / industry standard…
  - Will send in the group later on
- Questions, send via plain text instead of screenshot

Conditional expressions

The consequent and alternative expressions of conditional expressions are either in the same line as the predicate (if they fit), or they are aligned under the beginning of the predicate. Examples:

```
// good style
function abs(x) {
    return x >= 0 ? x : - x;
}

// good style
function abs(x) {
    return x > 0
           ? x
           : x === 0
           ? 0
           : - x;
}

// good style
const aspect_ratio = landscape ? 4 / 3 : 3 / 4;

// bad style: wasted lines
const aspect_ratio = landscape
                     ? 4 / 3
                     : 3 / 4;
```

Avoid parentheses around the predicate. Note that conditional expressions have lower precedence than all other operators in Source (except the assignment operator =, so parentheses around the predicate are rarely needed.

Recap

Substitution Model

What Is It?

- A **mental model** to facilitate understanding for stateless programming
- Stateless?
  - Names declared will *always* hold that value (key-value pair)
  - `const command = “sedia”;`
  - `function attention { return command; }`
  - Reassign? ❌

Substitution Model

Applicative Order Reduction

- Replace eligible sub-expression with result
  - Requires the arguments to be in their most “simple/basic” form before the function is applied
  - Evaluate then apply
- Used in Source, JavaScript, and many more languages!

Substitution Model

Normal Order Reduction

- Perform substitution before finding the exact value of the arguments
  - “Normal-order languages delay the evaluation of arguments until the argument values are needed”
  - Apply then evaluate
  - More complicated
- Laziness in evaluation
  - Will be covered in following lectures (as well as CS2030S (:)

Substitution Model

Applicative Order Reduction vs Normal Order Reduction

- Source §1 & Source §2
  - Can be modelled using substitution
- Applicative order reduction and normal order reduction should yield the same result

- ```
  function foo(x) {
      return x * x;
  }
  ```
- ```
  function bar(x) {
      return foo(x) + 1;
  }
  ```
- ```
  bar(5 + 3);
  ```
  ```

Substitution Model

Substitution in Mission “Rune Trials”

- ```
  function transform_mosaic(r1, r2, r3, r4, transform) {
      return transform(mosaic(r1, r2, r3, r4));
  }
  ```
- What if we change the parameter name into a pre-declared function?
- ```
  function transform_mosaic(r1, r2, r3, r4, make_cross) {
      return make_cross(mosaic(r1, r2, r3, r4));
  }
  ```
- Does this `make_cross` refer to the pre-declared function `make_cross`?
- Let’s try!

Substitution Model

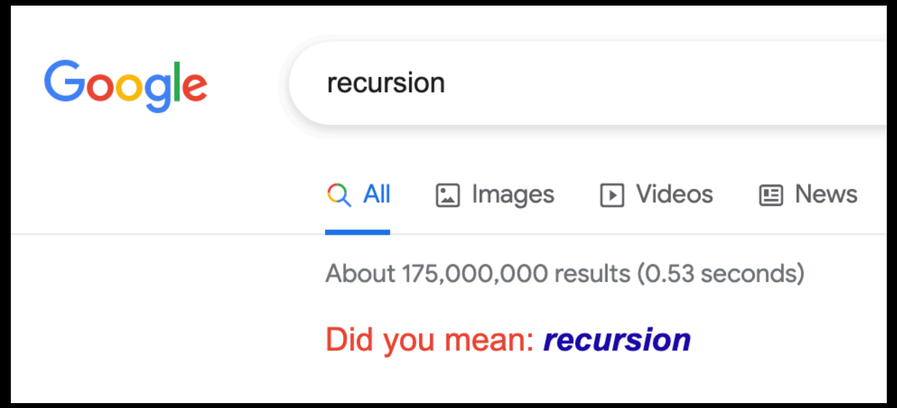
Substitution in Mission “Rune Trials”

- Parameter names are similar to constant declarations
- During declaration, we do not know what these constants are
- `make_cross` is now bound to `turn_upside_down` instead
- Naming in such a convention is a SUPER BAD practice
  - Feel free to rename function parameters to whatever seems more clear to you / your friends!

Recursion

What Is It?

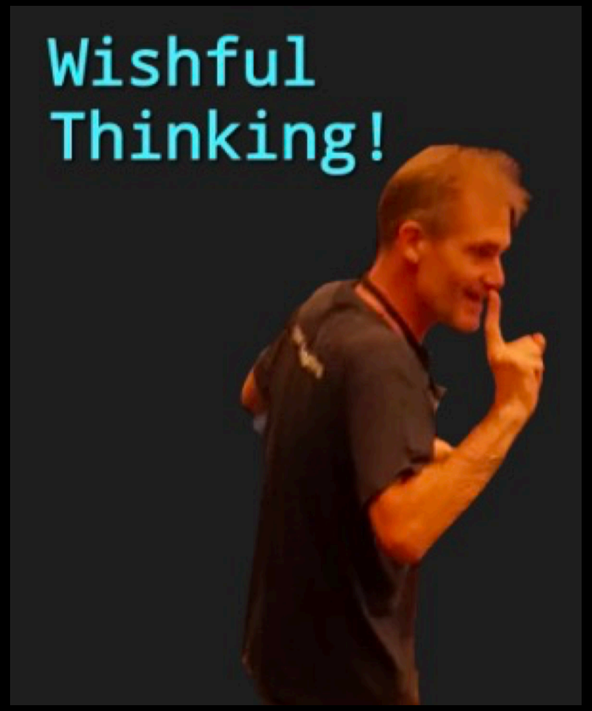
- “WISHFUL THINKING”
- Classic examples:
  - Factorial & Fibonacci sequence
- My own example:
  - Hawker centre queue



Recursion

Tips

- When solving questions pertaining to recursion:
  - Think about base case
    - Akin to CS1231S mathematical induction
    - Hook depth in the previous mission?
  - Imagine you have a magical wand
  - What would you do afterwards?
- Let’s try to practice them! Write `fact` and `fib` together



Recursion

Iterative Process vs Recursive Process

	Iterative	Recursive
Characteristic	More “diligent”	More “lazy”
Deferred operation	❌	✅
Space	Less	More
Analogy	Hardworking student who finishes work immediately	Lazy ass who dumps all the holiday assignments until the end to do at once

Recursion

Recursive Function vs Recursive Process

- Recursive Functions MIGHT NOT give rise to recursive process!
- Tail-call recursion
- Recursive process: look at deferred operations
- ```
  function fact(n) {
      return n === 1 ? 1 : n * fact(n - 1);
  }
  ```
- ```
  function fact_iter(product, counter) {
      return counter === 0
          ? product
          : fact_iter(counter * product, counter - 1);
  }
  ```

Recursion

Recursive Function vs Recursive Process

- Recursive function !== recursive process!!!

```
function fact(n) {
  return n === 1
    ? 1
    : n * fact(n - 1);
}
```

```
function fact_iter(product, counter) {
  return counter === 0
    ? product
    : fact_iter(
        counter * product,
        counter - 1
      );
}
```

Any Questions?
gotta rush ’em studio sheets…