NATIONAL UNIVERSITY OF SINGAPORE

CS1101S — PROGRAMMING METHODOLOGY

(AY2020/2021 SEMESTER 1)

READING ASSESSMENT 1

Time Allowed: **45 Minutes**

INSTRUCTIONS

1. This question paper comprises **NINE (9)** printed pages, including this page.

2. There are **16** multiple-choice questions. Each question has one correct answer. **1 mark** is awarded for each correct answer and there is no penalty for a wrong answer.

3. The full score is **16 marks**.

4. Answer **ALL** questions.

5. This is a **CLOSED BOOK** assessment, but you are allowed to refer to one A4 sheet of notes (handwritten or printed on both sides).

6. **Follow the instructions of your invigilator or the module coordinator to submit your answers**.

Scoping

(1) What is the result of evaluating the following Source program?

```
const u = 10;
const v = 20;

function foo(u) {
    const v = 4;
    return u + v + 3;
}

foo(u + 5);
```

- **A.** 38
- **B.** 33
- **C.** 22
- **D.** 17
- **E.** Error: one or more names is/are redeclared
- **F.** Error: one or more names is/are not declared before being used

(2) What is the result of evaluating the following Source program?

```
function f(w) {
    return 2 * g(w);
}

function g(x) {
    return w + x;
}

const w = 5;
f(w + 2);
```

- **A.** 20
- **B.** 24
- **C.** 28
- **D.** Error: names `g` and `w` are not declared before being used
- **E.** Error: only name `g` is not declared before being used
- **F.** Error: only name `w` is not declared before being used

(3) What is the result of evaluating the following Source program?

```
function f(w) {
    const x = 3;

    function g(x) {
        return w + x;
    }

    return 2 * g(w);
}

const w = 5;
f(w + 2);
```

A. 16
B. 20
C. 24
D. 28
E. Error: one or more names is/are redeclared
F. Error: one or more names is/are not declared before being used

(4) What is the result of evaluating the following Source program?

```
function f() {
    return y => z => x => w => 1000 * x + 100 * y + 10 * z + w;
}

f()(1)(2)(3)(4);
```

A. 1234
B. 4321
C. 4213
D. 3124
E. 3421
F. Error: wrong kind of arguments(s) or wrong number of argument(s)

(5) What is the result of evaluating the following Source program?

```
const x = 2;
const y = 3;

function foo(g, h, x, y) {
    return g(x) + h(y);
}

foo(x => x - y, y => x * y, 7, 4);
```

A. 5
B. 12
C. 15
D. 25
E. 31
F. Error: wrong kind of argument(s)

(6) What is the result of evaluating the following Source program?

```
const g = f => x => f(x * x);
g(x => x + 1)(4);
```

A. 17
B. 25
C. 26
D. 5
E. Error: wrong kind of argument(s)
F. Error: name `f` undeclared

(7) What is the result of evaluating the following Source program?

```
const x = 5;
const y = 2;

( (x, y)  =>  (y => y(x) + 1)(x => x * 3 + y) ) (x + 4, y + 1);
```

A. 21
B. 18
C. 30
D. 34
E. 31
F. Error: wrong kind of argument(s)

Processes

In some of the following questions, the pre-declared `display` function is used in the Source programs. The `display` function displays/prints the value of its input argument in the REPL. For example, `display(2 * 5)` prints `10`, and `display(1 > 2)` prints `false`.

(8) What is the sequence of values printed by the `display` function when the following program is evaluated?

```
function g(n) {
    display(n);
    return (n <= 1) ? n : n + h(n - 2);
}

function h(n) {
    display(n);
    return (n <= 1) ? n : n + g(n - 1);
}

g(10);
```

A. `10 8 7 5 4 2 1`
B. `10 9 7 6 4 3 1`
C. `10 8 6 5 4 3 1`
D. `10 9 8 6 4 2 1`
E. `10 8 7 6 4 2 1`

(9) What is the sequence of values printed by the `display` function when the following program is evaluated?

```
function f(x) {
    if (x === 0) {
        return x;
    } else {
        display(x);
        const y = 10 + f(x - 1);
        display(y);
        return y;
    }
}

f(4);
```

A. `4 3 2 1 10 20 30 40`
B. `4 3 2 1 40 30 20 10`
C. `4 10 3 20 2 30 1 40`
D. `4 40 3 30 2 20 1 10`
E. `4 11 3 21 2 31 1 41`

(10) What is the sequence of values printed by the `display` function when the following program is evaluated?

```
function fib(n) {
    display(n);
    return (n <= 1) ? n : fib(n - 2) + fib(n - 1);
}

fib(4);
```

A. 4 2 1 3 1 2 1

B. 4 2 3 0 1 1 2 0 1

C. 4 2 0 1 3 1 2 0 1

D. 4 3 2 1 0 1 2 1 0

E. 4 3 2 2 1 1 0 1 0

(11) What is the sequence of values printed by the `display` function when the following program is evaluated?

```
function fib(n) {
    if (n <= 1) {
        return n;
    } else {
        const x = fib(n - 2) + fib(n - 1);
        display(x);
        return x;
    }
}

fib(5);
```

A. 1 1 2 3 5

B. 5 3 2 1 1

C. 5 2 1 3 1 2 1

D. 1 2 1 3 1 2 5

E. 1 2 1 1 2 3 5

(12) What is the sequence of values printed by the `display` function when the following program is evaluated?

```
function D(m, x) {
    display(m);
    return x;
}
D(1, x => D(2, D(3, 3) * D(4, x))) (D(5, 5));
// same as (x => 3 * x)(5);
```

A. 1 2 3 4 5
B. 1 5 2 3 4
C. 1 5 3 4 2
D. 1 3 5 4 2
E. 1 3 4 5 2

(13) What kind of process does the following function `f` give rise to for any *integer* argument `n` > 0 and any integer argument `k`?

```
function f(n, k) {
    return n < 1
        ? k
        : f(n - 1, k + 1) + k;
}
```

A. An iterative process
B. A recursive process
C. A process that is both iterative and recursive
D. A substitution process
E. An infinite process

(14) What kind of process does the following function g give rise to for any *integer* argument `n` > 0?

```
function g(n) {
    function f(n, result) {
        return n < 1
            ? result
            : f( f(n - 1, 0), result + 1 );
    }
    return f(n, 0);
}
```

A. An iterative process
B. A recursive process
C. A process that is both iterative and recursive
D. A substitution process
E. A normal-order reduction process

Correctness

(15) We specify that the function *S*, when applied to two *integer* arguments, should return the sum of the arguments. Consider the following implementation:

```
function S(x, y) {
    return x === 0 ? y : S(x - 1, y + 1);
}
```

Which one of the following statements is correct?

A. The function `S` meets the specification.

B. The function `S` does not meet the specification because it can be applied to *non-integer* arguments.

C. The function `S` does not meet the specification because the argument `x` must always be larger than argument `y`.

D. The function `S` does not meet the specification because it is an inefficient way to compute the sum of two numbers.

E. The function `S` does not meet the specification because it does not work correctly for some valid arguments.

(16) We specify that the function *P*, when applied to any *positive integer* argument `n`, should return `true` if `n` is a prime number, and return `false` otherwise. A prime number is a positive integer that has exactly two factors (i.e. 1 and itself). Consider the following implementation:

```
function P(n) {
    function foo(d) {
        return d <= 1
            ? true
            : (n % d !== 0) && foo(d - 1);
    }
    return foo(n - 1);
}
```

Which one of the following statements is correct?

A. The function `P` meets the specification.

B. The function `P` does not meet the specification because it can be applied to *non-positive* and/or *non-integer* argument values.

C. The function `P` does not meet the specification because it does not check whether the argument is a valid input.

D. The function `P` does not meet the specification because its returned value is incorrect for some valid argument value(s).

E. The function `P` does not meet the specification because it is slow when the number argument is large.

——— END OF QUESTIONS ———

(blank page)