CS1101S Programming Methodology

Reading Assessment 1 from AY2017/18
adapted to CS1101S 2019/20

Use **only** the given answer sheet to indicate your answer to each of the following 15 questions. Use a pencil, and mark only one choice for each question. Do not write your name, but only your student number, on the answer sheet.

1 Scoping

Question 1:
What is the result of evaluating the following Source program:

```
const x = 75;
function f(y) {
    return x + 2;
}
f(x + 25);
```

1 [A] 25

1 [B] 75

1 [C] 77

1 [D] 100

1 [E] `Undeclared name x`

Answer: C: `x` in `x + 2` refers to 75, and thus the result is 77.

Question 2:
What is the result of evaluating the following Source program:

```
const x = 1;
function f(x) {
    return x + 2;
}
f(3);
```

2 [A] 3

2 [B] 4

2 [C] 5

2 [D] `Undeclared name f`

2 [E] `Undeclared name x`

Answer: C: `x` in `x + 2` refers to the argument `x` of `f`, and thus 3. Therefore the result is 5.

Question 3: What is the result of evaluating the following Source program:

```
function f(x) {
    return x => x + 1;
}
f(2)(3);
```

3 [A] 1

3 [B] 2

3 [C] 3

3 [D] 4

3 [E] `Undeclared name x`

Answer: D: `f(2)` evaluates to a function that adds 1 to its argument. That function is applied to 3, and therefore the result is 4.

Question 4: What is the result of evaluating the following Source program:

```
function f(g, x) {
     return g(g(x));
}
f(y => x + 1, 2);
```

4 [A] 4

4 [B] 3

4 [C] 2

4 [D] `Undeclared name g`

4 [E] `Undeclared name x`

Answer: E: The name `x` in `x + 1` is not in the scope of any name declaration.

Question 5: What is the result of evaluating the following Source program:

```
const x = 10;
function f(x) {
    function g(x) {
        return x + 20;
    }
    g(30);
    return x + 40;
}
f(50);
```

5 [A] 50

5 [B] 60

5 [C] 70

5 [D] 80

5 [E] 90

Answer: E: The name `x` in `x + 40` refers to the argument of `f`, and thus 50. So the result is 90.

2 Processes

Question 6: To what kind of process does the following Source program give rise to?

```
function f(x) {
    if (x < 10) {
        return x;
    } else {
        return f(x / 2);
    }
}
f(200);
```

6 [A] iterative process

6 [B] recursive process

6 [C] no process: there is a syntax error

6 [D] infinite process

6 [E] production process

Answer: A: The resulting process is a finite iterative process. Finite: Positive numbers larger than 10: The argument strictly decreases with each call. Other cases immediately lead to a result. Iterative: There are no deferred operations.

Question 7: To what kind of process does the following Source program give rise to when a positive integer is passed as argument?

```
function p(x) {
    if (x <= 10) {
        return p(x * 10);
    } else {
        return p(x - 1);
    }
}
p(200);
```

7 [A] no process: there is a syntax error

7 [B] recursive process

7 [C] illegal process

7 [D] infinite process

7 [E] production process

Answer: D: There is no syntax error. There are no deferred operations, therefore the process is not recursive. There is no legality problem (whatever that would mean). The term "production process" is not used in the module. For positive arguments larger then 10, the argument of `p` decreases until it reaches 10. After that, it jumps to 100, and the process continues. So the loop is infinite. For positive integers larger than 10, the value is multiplied by 10, after which the same argument as above holds. In both cases, an infinite loop ensues.

Question 8: To what kind of process does the following Source program give rise to?

```
function g(x) {
    if (x < 10) {
        return x;
    } else {
        return g(x / 2) * 2;
    }
}
g(200);
```

8 [A] iterative process

8 [B] recursive process

8 [C] no process: there is a syntax error

8 [D] infinite process

8 [E] production process

Answer: B: The process is recursive; the multiplication by 2 is a deferred operation that needs to be completed after the recursive call `g(x / 2)` returns.

Question 9: To what kind of process does the following Source program give rise to?

```
function j(x) {
  return k(x) - 1;
}
function k(x) {
  if (x === 0) {
    return 1;
  } else {
    return j(x - 1);
  }
}
k(200);
```

9 [A] iterative process

9 [B] recursive process

9 [C] no process: there is a syntax error

9 [D] infinite process

9 [E] production process

Answer: B: In each call of `k`, the function `j` is called, which according to the substitution model leaves behine a deferred operation (subtraction).

Question 10: To what kind of process does the following Source program give rise to?

```
function is_even(x) {
    return x % 2 === 0;
}

function q(x) {
  if (x <= 1) {
    return 0;
  } else if (is_even(x)) {
    return q(x / 2) + 1;
  } else {
    return q(x - 1);
  }
}
q(200);
```

10 [A] iterative process

10 [B] recursive process

10 [C] no process: there is a syntax error

10 [D] infinite process

10 [E] production process

Answer: B: The second case of the conditional has a deferred operation, so the process must be called *recursive*.

3 Correctness

Question 11: We specify that a function *zero* should always return the number 0 when applied to any argument value. Consider the following implementation.

```
function zero(x) {
  return "zero";
}
```

Which one of the following statements is correct?

11 [A] The function `zero` does not meet the specification for any argument.

11 [B] The function `zero` meets the specification only for some arguments.

11 [C] The function `zero` meets the specfication.

11 [D] The program does not define a function `zero`.

11 [E] The program has a syntax error.

Answer: A: The return value of the function `zero` is a string, but the specification requires a number.

Question 12: Recall that there are only two boolean values, `true` and `false`. We are specifying that the function `not` should be applied to a boolean value `b` and return a boolean value that is not `b`. Consider the following implmentation.

```
function not(b) {
    if (b) {
      return false;
    } else {
      return b;
    }
}
```

Which one of the following statements is correct?

12 [A] The function `not` does not meet the specification for any argument.

12 [B] The function `not` meets the specification only for the argument `true`.

12 [C] The function `not` meets the specfication only for the argument `false`.

12 [D] The function `not` meets the specfication.

12 [E] The program has a syntax error.

Answer: B: The result is correct for the argument `true`, but incorrect for the argument `false`.

Question 13: Consider the following specification of the triple function, defined on numbers:

$$triple(x) = 3x$$

For example, *triple*(10) is 30.
The following function is proposed as an implementation of *triple*.

```
function triple(n) {
    if (n === 0) {
      return 0;
    } else {
      return n + n + n;
    }
}
```

Which one of the following statements is correct?

13 [A] The function `triple` does not meet the specification for any argument.

13 [B] The function `triple` meets the specification only for the argument 0.

13 [C] The function `triple` meets the specfication only for non-zero arguments.

13 [D] The function `triple` meets the specfication.

13 [E] The program has a syntax error.

Answer: D: The program meets the specification. Its correctness is not affected by the unnecessary test `n === 0`.

Question 14: Consider the following specification of the tribonacci numbers T_i: $T_1 = 1, T_2 = 1, T_3 = 2$ and for any other positive integer n,

$$T_n = T_{n-1} + T_{n-2} + T_{n-3}$$

The following function `T` is proposed as an implementation of T.

```
function T(n) {
    if (n === 1) {
      return 1;
    } else if (n === 2) {
      return 1;
    } else if (n === 3) {
      return 2;
    } else {
      return T(n - 1) + T(n - 2) + T(n - 3);
    }
}
```

Which one of the following statements is correct?

14 [A] The function `T` does not meet the specification for any argument value.

14 [B] The function `T` meets the specification only for the numbers 1, 2 and 3.

14 [C] The function `T` meets the specfication.

14 [D] The function `T` meets the specfication for even argument values, but not for odd argument values.

14 [E] The program has a syntax error.

Answer: C: This is the direct translation of the specification to a program. Its correctness is not affected by the fact that it is unnecessarily time-consuming for larger values of `n`.

Question 15: Which one of the following statements is **false**?

15 [A] For a given specification, there can be two correct implementations.

15 [B] A specification can be written in English, with no formula.

15 [C] An implementation is considered correct if it meets the specification for most of the specified values.

15 [D] An implementation can be correct even if it is unnecessarily complicated.

15 [E] An implementation can be correct even if it is slower than other correct implementations.

Answer: C: For an implementation to be correct, it needs to meet the specification for all specified values, not just "most of" them.