CMPT-166: Sample Midterm

Last name *exactly as it appears on your student card*	**First name** *exactly as it appears on your student card*

Student Number									
SFU Email		**Section** if you know it!	
---	---	---	---

This is a **50 minute** test. It is **closed book**: no calculators, computers, notes, books, etc. are allowed.

True or False questions: fill in the corresponding T or F bubble for that question on your answer sheet.

Multiple Choice questions: fill in the corresponding letter bubble on your answer sheet.

Correct answers are worth 1 mark, and incorrect or illegible answers, or unanswered questions, are worth 0.

Question	Out Of
True or False	50
Multiple Choice	15
Total	**65**

Consider the following Processing program::

```
void setup() {
    size(300, 200);
}

void draw() {
    rect(mouseX, mouseY, 10, 20);
}
```

1.	When the program runs, `setup()` is called exactly once.
2.	By default, `draw()` is called about 60 times per second.
3.	If the `setup()` function was put **after** the `draw()` function, then, when the program runs, `draw()` would be called first, and `setup()` would be called second.
4.	The drawing screen is 200 pixels wide.
5.	There are no source code comments.
6.	The program would still compile if all the `;` characters were deleted.
7.	The program would still compile, and run the same, if all the `{` characters were put on their own line.
8.	The rectangle that appears on the screen leaves a trail when it moves.
9.	(`mouseX`, `mouseY`) is the center of the rectangle that is drawn.
10	The height of the rectangle is 10 pixels.

Consider the following fragment of Processing code:

```
void setup() {
   int a = 2;
   a += 3;
   int b = 1;
   b += a + b;
   println(a – b);
}
```

11.	There are exactly three different `int` **variables** defined in this code fragment.
12.	There are exactly two different `int` **literals** in this code fragment.
13.	The statements `a += 3` adds 3 to `a`.
14.	When run, this code fragment prints -2.
15.	As far as the Processing compiler is concerned, `a += 3` is the same as `a + = 3` (space added between the + and =).
16.	As far as the Processing compiler is concerned, `a += 3` is the same as `a+=3` (spaces before and after += are deleted).
17.	If the statement int `b = 1` where changed to `int b = 2`, then this code fragment would print -3.
18.	If `a += 3;` was replaced by `// a += 3;`, then the program would **not** compile (i.e. there would be a compiler error).

19.	The program would still compile, and run (but may print different output), if the order of the first two statements were swapped like this:
	
	`a += 3;`
	`int a = 2;`

20.	`println` prints its output on the same window where a rectangle or ellipse would be drawn.

Consider the following Processing program:

```
PImage mascot;

void setup() {
  size(500, 500);
  mascot = loadImage("duke.png");
}

void draw() {
  background(255);
  image(mascot, mouseX, mouseY);
  image(mascot, mouseY, mouseX);
}
```

21. `mascot` is the name of a variable of type `PImage`.
22. The file "duke.png" will be loaded correctly if it is in a folder named **`images`** inside the folder for this program.
23. Processing cannot display JPEG images.
24. The background color of the drawing window is black.
25. The program would probably run faster if you move the line "`mascot = loadImage("duke.png");`" out of `setup()` and put it into `draw()` (just before the calls to `image`).
26. Exactly two copies of `mascot` are drawn on the screen, but you can only see one copy when you run the program because the images are always drawn on top of each other.
27. The center of the first image drawn is (`mouseX`, `mouseY`).

Consider the following fragment of Processing code:

```
int x = 3;  // initial value of x
int y = 3;  // initial value of y

if (x <= y) {
   println("hello!");
} else {
   println("goodbye!");
}
```

28.	It prints “hello!” when run.
29.	If `<=` were replaced by `>=`, the program would print “goodbye!” when run.
30.	If `<=` were replaced by `!=`, the program would print “goodbye!” when run.
31.	If `<=` were replaced by `<`, the program would print “goodbye!” when run.
32.	If `<=` were replaced by `>`, the program would print “hello!” when run.
33.	If `<=` were replaced by `==`, the program would print “hello!” when run.
34.	If `<=` were replaced by `=`, the program would print “hello!” when run.
35.	If `x <= y` were replaced by `x + 1 <= y - 2`, the program would print “goodbye!” when run.
36.	If `x <= y` were replaced by `x == x`, the program would print “hello!” when run, even if you changed the initial values of x and y to be randomly chosen `int`s.
37.	If the two `{` characters and the two `}` characters were deleted, then the code would still compile, and run the same way.

For each of the following boolean expressions, choose T if it evalautes to true, and F otherwise.

38.	`true || false`
39.	`true || true`
40.	`false && false`
41.	`false || false || true || false`
42.	`!(true || false)`
43.	`(!false) && true`
44.	`!!!true`
45.	`!(!false && !false)`
46.	`!(!false || !false)`
47.	`!(true && !!(false || false))`

48.	If `a` and `b` are both variables of type `int`, and `a` > 0 and `b` > 0, then it is always the case that `a + b > 0`.
49.	If `a` is a variable of type `int`, and `a < 0`, then it is always the case that `abs(a) > 0` (`abs` calculates the absolute value of `a`).
50.	If a is a variable of type int, then a – a is always 0.

Multiple Choice

51. Processing is an example of:

    a) a scripting language
    b) an assembly language
    c) a low-level language
    d) a high-level language

52. Which one of the following tokens marks the beginning of a code block in Processing?

    a) `{`
    b) `(`
    c) `//`
    d) `)`

53. Which of the following is a major difference between Processing's default screen coordinates and the coordinates used in mathematics?

    a) the x and y values of a coordinate are swapped, e.g. in math you would write (3, 2), but in Processing you would write it (2, 3)
    b) the x-axis increases to the left (instead of to the right)
    c) the y-axis increases down (instead of up)
    d) there is no difference

54. Processing's `quad` function is used to draw shapes with 4 edges, such as squares, rectangles, and bow-ties. How many different input numbers does quad require?

    a) 4
    b) 8
    c) 12
    d) 16

55. Approximately how many distinct colors can Processing's RGB color notation represent?
    a) 16.7 thousand
    b) 16.7 million
    c) 16.7 billion
    d) 16.7 trillion

56. What number should replace `?` to make this statement print 5.0?

```
println(map(?, 0, 20, 0, 10));
```

    a) 0
    b) 5
    c) 10
    d) 20

57. What does CPU stand for?
    a) computer power unit
    b) computer processing unit
    c) central power unit
    d) central processing unit

58. How many different levels of transparency does Processing's RGB color support?
    a) 0
    b) 2
    c) 256
    d) about 16 million

59. What does this print?

```
print(min(max(3, 2), min(4, 7)));
```

    a) 2
    b) 3
    c) 4
    d) 7

60. Which one of these statements is **false**?

    a) folders can contain folders or files (and maybe other things, like shortcuts)

    b) a folder cannot contain 2 (or more) things with the same name (except for one special case!)

    c) in Processing, absolute path names are almost always better and more useful than relative path names

    d) folder and directory mean the same thing

61. What does this print?

```
void setup() {
   int n;
   println(n);
}
```

    a) nothing: it causes a compile-time error

    b) nothing: it causes a run-time error

    c) 0

    d) some unknown integer value

62. What does this print?

```
void setup() {
   println(n);
}

int n;
```

    a) nothing: it causes a compile-time error

    b) nothing: it causes a run-time error

    c) 0

    d) some unknown integer value

63. What does this print?

```
void setup() {
   float x = int(3.0);
   println(x);
}
```

a) nothing: it causes a compile-time error

b) nothing: it causes a run-time error

c) 3

d) 3.0

64. In Processing, an `int` is represented by 32 bits. What is the maximum positive number that a Processing `int` can represent?

a) $2^{31}-1$

b) 2^{31}

c) $2^{32}-1$

d) 2^{32}

65. Which one of the following is a string literal in Processing?

a) `apple`

b) `'apple'`

c) `"apple"`

d) none of the above