UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

Department of Computing

B. Sc. Examination 2018

IS50001C Foundations of Programming

Duration: 2 hours 15 minutes

Date and time:

This paper is in two parts: part A and part B. Part A carries 40 marks, and each question from part B carries 30 marks. ***You should answer ALL questions from part A and TWO questions from part B.*** *If you attempt all 3 questions in part B, only the* ***first two*** *that you attempt will be marked. The marks for each part of a question are indicated at the end of the part in [.] brackets.*

There are 100 marks available on this paper.

You are not allowed to use any mobile device (such as telephones, calculators, tablets) during the exam.

THIS PAPER MUST NOT BE REMOVED
FROM THE EXAMINATION ROOM

Part A

You should attempt all of these questions

Each multiple-choice question has one (and only one) correct answer.
For each question, write your choice on your *answer* book.

(a) What is the final value of variable `y` after executing the following code?

```
y = int("2")
x = y-1
y = x
```

i. -1
ii. 1
iii. 2
iv. None of the above: the code generates a runtime error.

[2]

(b) What is printed on the screen when the following lines of Python code are executed?

```
a = 0
b = 2
print('(a-b) is', str(a-b))
```

i. A syntax error
ii. The string `"(a-b) is str(a-b)"`
iii. `(a-b) is -2`
iv. Nothing on the screen; the local printer will print "`(a-b) is str(a-b)`"

[2]

(c) What is the final value of the variable `x` after executing the following assignment:

```
x = 0.0 - 2
```

i. `2`
ii. `-2.0`
iii. `-2`
iv. `0.0`

[2]

(d) What is produced on the screen by the 2 lines of Python code below?

```
a = "1"
print(float(a))
```

i. An error message.
ii. `1.0`
iii. `1`
iv. The string `"float(a)"`.

[2]

(e) What is the value of variable `c` after executing the following code?

```
b = 25
c = b%5
```

i. 0
ii. 5
iii. The value of `c` will be undefined, as the second assignment contains a syntax error.

[2]

(f) What does execution of the following code snippet produce on the screen?

```
x = False
print(x and 4/2 == 2)
```

i. `False`
ii. `True`
iii. The string `"x and 4/2 == 2"`.
iv. An error message.

[2]

(g) Which message will the following Python statements print on the screen?

```
n = 1
m = 0
if n > m:
      print("The value", n, 'is greater than 0.')
else:
      print("The value", n, 'is less than 0.')
```

i. `The value 0 is less than 0.`
ii. `The value 1 is less than 0.`
iii. `The value 1 is greater than 0.`

[2]

(h) What will the following lines of Python code produce?

```
var1 = 5
print("The value of 'var1' is less than 0.")
```

i. A syntax error.
ii. The message "`The value of 5 is less than 0.`"
iii. The message "`The value of 'var1' is less than 0.`"

[2]

(i) Does the following code excerpt contain a semantic error?

```
x = 5//2
print(x, "is less than 0.")
```

i. Yes.
ii. No.

[2]

(j) Which line of code must be executed *before* the one below to avoid a runtime error?

```
tess.color('red')
```

i. `tess = color.Turtle()`
ii. `import Screen()`
iii. `tess = turtle.Turtle()`
iv. `import random`

[2]

(k) The correct command to generate a random float between 2 and 3 (excluding 3) is:

i. `2/3 + random.random()`
ii. `2 + 3*random.random()`
iii. `2 + random.random()`
iv. `2 - random.random()`

[4]

(l) Assuming that the user types in "2.0" (followed by the *Enter* key) when prompted for a number, what is produced on the screen by the following 2 lines of Python code?

```
x = input("Please enter a number: ")
print(x > 1)
```

i. The string `"x > 1"`.
ii. The string the user entered (i.e., `"2.0"`).
iii. `True`
iv. An error message

[4]

(m) How many times will the following ***for*** loop print the message "Hello" on the screen?

```
n = 2
for i in range(n*n, n-1, -1):
    print("Hello")
```

i. 2 times
ii. 3 times
iii. 4 times
iv. None of the above.

[4]

(n) What does the following code snippet produce in output?

```
s = ""
for i in "string":
   s = i+s
print(s)
```

i. `string`
ii. `gnirts`
iii. `gnirtsstring`
iv. The empty string (`""`), because the *for* loop fails to execute.

[4]

(o) What does the following Python program print?

```
def myCase(x):
   if x > 0:
        return "Case " + str(x)
   elif x == 0:
        return "Case " + str(x)
   else:
        return "All other cases"

x = 0
print(myCase(-1))
```

i. `Case 0`
ii. `Case -1`
iii. `All other cases`
iv. None of the above, because the program contains an error.

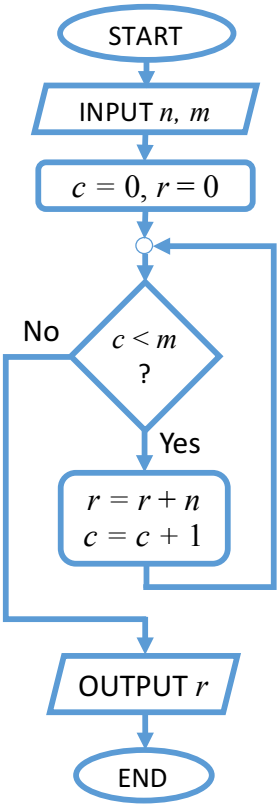
[4]

Part B

You should attempt two of these three questions

QUESTION B1

(a) Consider the algorithm described by the flowchart below, with *n, m* integers and ≥ 0.



i. What happens to the variable *c* in the loop body? And to *r*?

[2]

ii. How many times will the loop body be repeated in the following 3 different cases: $m = 1$, $m = 2$, $m = 3$?

[3]

iii. So, in general, for any given value of *m*, how many times is the loop body repeated?

[2]

iv. In view of the above, what is the value of *r* printed at the end, for any given values of *n, m* provided by the user?

[3]

v. Write a Python program that implements this algorithm using a ***while*** loop (Note: your code is not required to contain a new function definition).

[10]

(b) The following code extract is meant to get the user to enter an integer > 0 and store it in variable `x`:

```
1  x = -1
2  while x < 0:
3  x = str(input("Enter an integer smaller than 0:")
```

i. What is wrong with this code? Indicate what should be changed, and how, to make it work correctly.

[6]

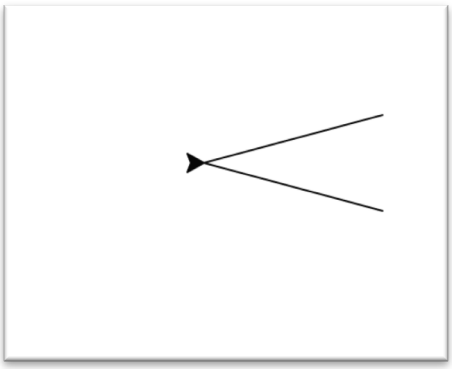
ii. For each error you listed in your answer to point i. above, indicate whether it is a syntax, runtime, or semantic error.

[4]

QUESTION B2

(a) Reorder the following lines of code so that, after drawing, the window is identical to the figure shown below. The turtle, called "`jo`", should complete the drawing back in its initial position of (0,0) pointing "East" (Note: the internal angle between the two lines is 30°):

```
0  jo.forward(100)
1  jo.left(15)
2  import turtle
3  jo.forward(100)
4  wn = turtle.Screen()
5  jo.backward(100)
6  jo.backward(100)
7  jo.left(15)
8  jo = turtle.Turtle()
9  jo.right(30)
```



[6]

(b) Define a new function, called `drawV(t,sz)`, which takes a turtle `t` and a number `sz` as input parameters, and draws the figure of question **(a)** above, having two segments each `sz` pixels long. At the end of the function, the turtle should end up in the same position (and point in the same direction) as at the beginning of the function.

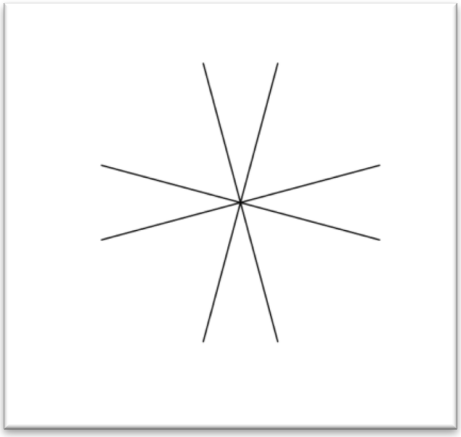
For example, the command `drawV(jo,100)` should draw exactly the same figure on the canvas as the one shown in question **(a)** above.

Hint: the body of the function '`drawV`' will contain many of the commands listed above, but should make use of the two parameters `t` and `sz` where appropriate.

[8]

(c) Write a program that calls the function `drawV(t,sz)` (written for point **(b)** above) four times, to draw the figure shown below:

[8]



(d) Consider the following lines of Python code:

```
1   import turtle
2   wn = turtle.Screen()
3   jo = turtle.Turtle()
4   jo.color('blue')
5   jo.forward(50)
6   tess = jo
7   tess.color('red')
8   tess.left(90)
9   jo.forward(50)
```

i. What happens when the above commands are executed? Draw a simple diagram showing what the graphic window (canvas) will look like at the end, naming the colour of any lines you draw in it.

[4]

ii. Briefly describe what happens in Python's internal representation of variables and objects when line 6 above – `tess = jo` – is executed.

[4]

QUESTION B3

(a) Consider the following code fragment, containing 2 nested ***for*** loops:

```
x = []
for i in range(1,3,1):
     for j in [1,3,1]:
         x.append(i+j)
```

i. Which list of integers does the (*outer*) loop variable `i` iterate over?

[2]

ii. Which list of integers does the (*inner*) loop variable `j` iterate over?

[3]

iii. How many times is the instruction "`x.append(i+j)`" executed, in total?

[2]

iv. When `i=1` and `j=1`, what is the effect of instruction "`x.append(i+j)`" on `x`?

[3]

v. In view of the above, what does `x` contain after the two ***for*** loops have completed?

[5]

(**b**) Write a Python function called `shared(x, y)` that takes two strings `x`, `y` as input parameters and returns the number of characters in string `x` that also appear in `y` (in other words, the number of characters that string `x` shares with string `y`).

> **For example**, `shared('Rubber', 'bear')` should return 4, as each of the last four characters of `'Rubber'` appears in the string `'bear'`. Note that `'R'` (in uppercase) does not appear in `'bear'`).

[15]

End of Exam