

**UNIVERSITY OF LONDON**

**GOLDSMITHS COLLEGE**

**Department of Computing**

**B. Sc. Examination 2020**

**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 electronic device (such as mobile telephones, laptops, 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.**

(1) In the following Python code, what value is assigned to the variable `x`?

```
x = 2**4
```

- a. 8
- b. 16
- c. 24
- d. No value, the code causes an error

[2]

(2) What do the following lines of code produce on the screen?

```
a = 1  
b = 1  
print('a*b =', float(a-b))
```

- a. The message “1 = 0”
- b. The message “a\*b = 0.0”
- c. The message “a\*b = 0”
- d. Nothing on the screen; the local printer will print “a\*b = float(a-b)”

[2]

(3) What is displayed on the screen when the 2 lines of code below are executed?

```
s = "173"  
print(len(s))
```

- a. 1
- b. 3
- c. 173
- d. An error message

[2]

(4) What does the following code produce on the screen?

```
y = str(1)  
x = y - 1.0  
print(x)
```

- a. 0
- b. 1
- c. 0.0
- d. An error message

[2]

(5) What does the following line of Python code produce as output?

```
print(False and True)
```

- a. The string “False and True”
- b. False
- c. True
- d. An error message

[2]

(6) What is the final value of variable `y` after the following statements are executed?

```
x, y = 0, 1
y *= -2
y, x = x, y
```

- a. 0
- b. -2
- c. 1
- d. None of the above.

[2]

(7) Which of the statements following the code below is true? Choose only one answer:

```
x = 1
for i in range(x, 10)
    print(i)
```

- a. The code contains no errors
- b. The code contains a syntax error
- c. The code causes a runtime error
- d. The code contains a semantic error (which does not cause an error message)

[2]

(8) What type will variable `n` be after the following lines have been executed?

```
s = "Please enter a float number larger than 0:"
n = input(s)
```

- a. String
- b. Float
- c. Integer
- d. It depends on what the user enters

[2]

(9) In the following code, what must `x` be assigned to for the “if” clause to be executed?

```
x = ...
if (x%5) == 0:
    print("The condition has been met")
```

- a. 0
- b. A multiple of 5
- c. 0 or a multiple of 5
- d. The “if” clause will never be executed

[2]

(10) How many times will the string “Hello” be printed when the following code is run?

```
for i in [0, 3, 1]:
    print("Hello")
```

- a. 1 time
- b. 2 times
- c. 3 times
- d. 0 times (because the code generates a runtime error)

[2]

---

(11) What value is assigned to variable x by the following code?

```
x = 834.2 // 81.3
```

- a. 10
- b. 10.0
- c. 10.2607626076...
- d. None of the above, there is an error

[4]

(12) What is the on-screen output produced by the following code?

```
s = "Yes!"  
print(s[:1] + s)
```

- a. YYes!
- b. YeYes!
- c. Yes!s
- d. An error message

[4]

(13) What does execution of the following *while* loop produce on the screen?

```
x, y = 0, 0  
while x < 3:  
    x, y = x+1, y+2  
print(y)
```

- a. 2
- b. 4
- c. 6
- d. The execution of the loop body produces an error

[4]

(14) We wish to generate a random float between 2 and 4 (excluding 4). Which of the following commands should we use?

- a. `2 + random.random()`
- b. `4*random.random() - 2`
- c. `4 - random.random()`
- d. `2 + 2*random.random()`

[4]

(15) What does the following line of code produce on the screen?

```
print([3, 2, 1] + [6, 5, 4])
```

- a. [3, 2, 1, 6, 5, 4]
- b. [9, 7, 5]
- c. [1, 2, 3, 4, 5, 6]
- d. None of the above

[4]

## Part B

You should attempt two of these three questions

## QUESTION B1

(a) Consider the following function definition:

```
def mystery(n):  
    temp = 0  
    for i in range(n):  
        temp = temp + 2*(i+1)  
    return temp
```

i. What values does `mystery(n)` return for  $n=1$ ,  $n=2$ , and  $n=3$ ?

[3]

ii. What do you think it returns in general (i.e., for any number  $n$ )?

[4]

(b) What is wrong with the following program? Provide a *brief* explanation, indicating what should be added to or changed in the code to remove any error(s).

```
1 def isDivisible(x,y):  
2     if x%y == 0:  
3         result = "Yes"  
4     else:  
5         result = "No"  
6  
8 print(isDivisible(12,3))
```

[6]

(c) The function “!” (factorial) is defined as follows:

*Given a positive integer  $n$ ,  $n!$  is the number obtained by multiplying all the whole numbers from 1 to  $n$ ;  $0!$  is defined to be 1.*

For example,  $3!$  is 6, since  $1*2*3 = 6$ . Note that  $1! = 1$  (and, by definition,  $0! = 1$ ).

Write a new Python function called `factorial(n)` that takes an integer  $n$  as input (assume  $n > 0$ ) and returns the value  $n!$  (i.e.,  $n$  factorial).

**HINT:** use a *for* or a *while* loop.

[7]

- (d) Suppose that the new release of Python no longer includes the *for* statement. Rewrite the following program using only *while* statements (and no *for* loops), ensuring that the output produced remains exactly the same:

```
for i in range(3):
    print(i)
    for j in range(5):
        print(j)
```

[10]



## QUESTION B2

(a) Consider the following code, in which the lines have been numbered for convenience:

```
1 import turtle
2 def moveOn(t, length):
3     t = turtle.Turtle()
4     t.forward(length)
5
6 alex = turtle.Turtle()
7 alex.color("red")
8 moveOn(alex,100)
```

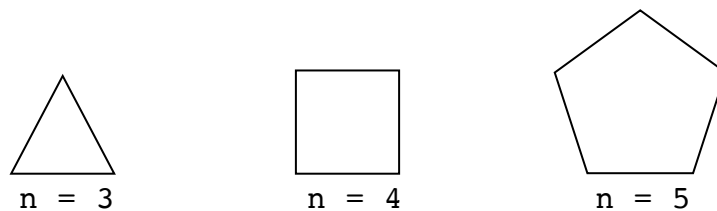
- i. What value does the function “moveOn ( )” return? [2]
- ii. How many (formal) parameters does moveOn ( ) have, and what are they called? [2]
- iii. Assuming that the order in which the above lines are executed is 1, 2, 6, 7, 8, 3, 4, at which of these steps are the parameters of the “moveOn ( )” function assigned to specific values / objects? What are they assigned to? [2]
- iv. What colour is the segment drawn on the canvas when line 4 is executed? [3]
- v. Where will the turtle object named ‘t’ be, on the canvas, at the end of function “moveOn ( )”? Indicate its exact final (x, y) co-ordinates. [3]
- vi. At which location on the canvas will the turtle object named ‘alex’ be after line 8 has been executed? Draw a diagram and provide its final (x,y) co-ordinates. [4]
- vii. Suppose that line 3 of the above program was deleted. Would your answer to question iv. above (“*What colour...*” etc.) change? If so, how? If not, why not? [4]

- (b) The code below defines a new function called **drawPolygon**(*t*, *n*, *sz*) which uses a turtle *t* to draw a regular polygon of *n* sides ( $n > 2$ ), each side *sz* pixels long (recall that the external angles of a regular polygon are all identical and equal to  $360^\circ/n$ ):

```
import turtle

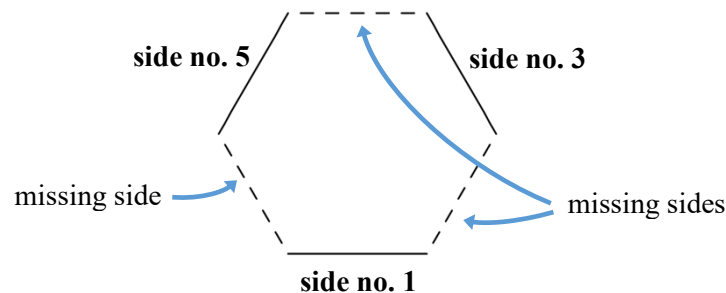
def drawPolygon(t,n,sz):
    angle = 360/n
    for i in range(n):
        t.forward(sz)
        t.left(angle)
```

For example, assuming *t* is a turtle, drawPolygon(*t*, **3**, 20) draws an equilateral triangle of size 20; drawPolygon(*t*, **4**, 20) draws a square; drawPolygon(*t*, **5**, 20) draws a regular pentagon, etc. (see examples below).



- i. Write a new function, called **drawOddSides**(*t*, *n*, *sz*), which uses the turtle *t* to draw *only* the “odd numbered” sides of a regular *n*-sided polygon ( $n > 2$ ) of size *sz*. Assume the lowest horizontal segment is side no. 1 (see example figure below).

For example, if  $n = 6$ , the figure drawn should only contain sides number 1, 3 and 5 of a regular hexagon (i.e., the solid lines in the diagram below):



**HINT:** think of how the drawPolygon function could be changed to achieve this..

[10]

### QUESTION B3

- (a) The following Python function takes a string `s` as input and returns either `False` or `True` depending on the result of a check it carries out on `s`:

```
def checkString(s):
    i = 0
    j = len(s)-1
    while i<j:
        if s[i] != s[j]:
            return False
        i = i+1
        j = j-1
    return True
```

- i. What does the function return when it is passed an empty string? And when it is given a string containing just 1 character? [2]
  - ii. How many times is the body of the *while* loop executed when the function is passed the string "ABA"? And when it is given the string "ABBA"? [2]
  - iii. What values would `checkString` return for strings "ABA" and "ABBA"? [2]
  - iv. Would the answer to point iii. above change if the strings given as input to the function were "xZx" and "xZZx" instead of "ABA" and "ABBA"? [2]
  - v. In view of all of the above, what do you think the function checks / returns, in general? [4]
- (b) Write a new Python function called `overlap(x, y)` that takes two lists `x`, `y` of integers and returns a new list containing all (and only) the numbers that appear in *both* `x` and `y` (the order is unimportant). The list returned should contain no duplicates. If either of `x` or `y` is empty, or if `x` and `y` share no elements, the function should return the empty list, `[]`.

Examples:

`overlap([1, -5], [1])` should return `[1]`

`overlap([3, 5, -1, 5], [1, 76])` should return `[]`

`overlap([2, 43, 2, -9], [-9, 0, 2])` should return `[2, -9]` (or `[-9, 2]`)

(In the last example, note that the number 2 appears only *once* in the result).

[8]

- (c) Consider your answer to point (b) above: how would your code need to be changed for it to work also with lists containing elements other than integers (i.e, floats, bools, characters, or a mix of them)? Provide a *short* explanation.

[4]

- (d) Consider the following incomplete definition of a class for University students:

```
class student:
    def __init__(self, ...):
        ...
```

Complete the above class definition, replacing the “...” with the constructor body and any parameters needed, as appropriate. When creating a particular student, it should be possible to specify name, student id, and credits accumulated. When a new student object is created, if the credits are left unspecified, they should be set to 0 by default.

[6]

## End of Exam