COMP 102 — PROGRAMMING FUNDAMENTALS FORMAN CHRISTIAN UNIVERSITY · FALL 2026

Quiz 04 — Big-O, Dictionaries & Exceptions

Scope: Lectures 15–18 only · **Time:** 10 minutes · **Marks: 10**

Name: ..... **Roll no:** .....

Section: A **Date:** Nov 10, 2026

(1) Big-O [3 marks]

Write the Big-O of each function. `L` is a list of `n` items. (1 mark each)
Choose from: `O(1)` `O(log n)` `O(n)` `O(n^2)` `O(2^n)`

(a)

```
def f(L):
    total = 0
    for x in L:
        total = total + x
    for k in range(50):
        total = total + 1
    return total
```

(b)

```
def g(n):
    steps = 0
    size = 1
    while size < n:
        size = size * 2
        steps = steps + 1
    return steps
```

(c)

```
def h(L):
    for x in L:
        print(x, L.index(x))
```

(a) ..... (b) ..... (c) .....

(2) Trace [2 marks]

A pet shop counts its animals. What does this program print? Write the **exact** output, one line per line. (1 mark per line)

```
1 pets = {"cat": 3, "dog": 1}
2 pets["fish"] = 2
3 pets["cat"] = pets["cat"] + pets["dog"]
4 del pets["dog"]
5 print(pets)
6 print(len(pets), 2 in pets)
```

.....

.....

(3) Fill the blank [2 marks]

`passed(marks)` returns a **new** dictionary with only the students who got 50 or more.
`passed({"ali": 72, "sana": 45, "omar": 50})` returns `{"ali": 72, "omar": 50}`. (1 mark each)
Write the missing code for each blank on the line below with the same letter.

```
1  def passed(marks):
2      result = {}
3      for ___ⓐ___ :
4          if marks[name] >= 50:
5              ___ⓑ___
6      return result
```

ⓐ ..... ⓑ .....

(4) Value or error? [3 marks]

A square garden has an area. `side(area)` finds the length of one side. For each call, write the value it returns, or the **name** of the error it raises. (1 mark each)

```
1  def side(area):
2      assert area >= 0, "area cannot be negative"
3      guess = 0
4      while guess * guess < area:
5          guess = guess + 1
6      if guess * guess != area:
7          raise ValueError("not a square")
8      return guess
```

(a) `side(49)` ..... (b) `side(50)` ..... (c) `side(-4)` .....

Rough work (not marked)

Quiz 04 — Big-O, Dictionaries & Exceptions

Scope: Lectures 15–18 only · **Time:** 10 minutes · **Marks: 10**

Name: ..... **Roll no:** .....

Section: B **Date:** Nov 10, 2026

(1) Big-O [3 marks]

A program takes 3 seconds on 1000 names. Now we give it 2000 names. About how many seconds does it take now? (1 mark each)

(a) The program is `O(n)`: ..... seconds (b) The program is `O(n^2)`: ..... seconds

(c) `L` is a list of `n` numbers. Write the Big-O of `above_avg(L)`.
Choose from: `O(1)` `O(log n)` `O(n)` `O(n^2)` `O(2^n)`

```
def above_avg(L):
    avg = sum(L) / len(L)
    count = 0
    for x in L:
        if x > avg:
            count = count + 1
    return count
```

(c) .....

(2) Trace [2 marks]

Students are put in teams. What does this program print? Write the **exact** output, one line per line. You may not need every line. (1 mark per line)

```
teams = {"red": ["ali"], "blue": ["sana", "omar"]}
teams["red"].append("zara")
teams["green"] = ["hina"]
for t in teams:
    if len(teams[t]) > 1:
        print(t)
```

.....

.....

.....

③ Fill the blank **[2 marks]**

A shop adds each day's sales to the week's total.
`week = {"pen": 5}`. After `add_sales(week, {"pen": 2, "ink": 3})`, `week` is `{"pen": 7, "ink": 3}`. (1 mark each)
Write the missing code for each blank on the line below with the same letter.

```
1 def add_sales(total, day):
2     for item in day:
3         if item in total:
4             total[item] = ___ⓐ___
5         else:
6             total[item] = ___ⓑ___
```

ⓐ ..... ⓑ .....

④ Search and floats **[3 marks]**

(a) Sara picks a secret whole number from 1 to 19. The program starts with `low = 0` and `high = 20`. Each guess is `(low + high) // 2`. If the guess is too low, `low` becomes the guess. If it is too high, `high` becomes the guess. Sara's number is 16. The first guess is 10. Write the next guesses, in order. The last one finds 16. (2 marks)

Guess 1: 10 Guess 2: ..... Guess 3: ..... Guess 4: .....

(b) What does this program do? Write what it prints, or the **name** of the error it raises. (1 mark)

```
1 x = 0.7 + 0.1
2 assert x == 0.8, "x is not 0.8"
3 print("done")
```

(b) .....

Rough work (not marked)