Worksheet 18: Time Complexity

Name: ____________________ Date: ____________________

Instructions

- Answer in the blanks.
- For “time complexity” questions, choose from: **O(1)**, **O(n)**, **O(n²)**.
- `n` means the input size (for lists: `n = len(numbers)`, for strings: `n = len(s)`).
- For “what does it print” questions, write the **exact** output (line by line).

Part A — Time complexity basics

1) Fill in the blanks (meaning)

Complete the sentences:

- Time complexity describes how the amount of work grows when the input size _____. (grows / shrinks)
- **O(1)** means the program does about the _____ number of steps no matter what `n` is. (same / different)
- When we talk about time complexity, we usually report the _____ case. (best / worst)

2) What is the input size `n`?

For each example, write what `n` is.

1. `numbers = [2, 5, 1, 4]` → `n = ______________________`
2. `s = "ABCDE"` → `n = ______________________`

Part B — Classify code as O(1), O(n), or O(n²)

3) First item (no loop)

```
numbers = [2, 5, 1, 4]
print(numbers[0])
```

Time complexity: ____________________

---

4) Swapping numbers

```
a = 6
b = 7

temp = a
a = b
b = temp
```

Time complexity: ____________________

---

5) Print every item (one loop)

```
numbers = [2, 5, 1, 4]

for x in numbers:
    print(x)
```

Time complexity: ____________________

---

6) Two loops in a row (still linear)

```
numbers = [2, 5, 1, 4]

for x in numbers:
    print(x)

for x in numbers:
    print(x * 2)
```

Time complexity: ____________________

---

7) Pairwise print (nested loops)

```
numbers = [2, 5, 1, 4]
n = len(numbers)

for i in range(n):
    for j in range(n):
        print(i, j)
```

Time complexity: ______________________

8) Inner loop runs a constant number of times

```
numbers = [2, 5, 1, 4]
n = len(numbers)

for i in range(n):
    for _ in range(5):
        print(i)
```

Time complexity: ______________________

9) Triangle nested loops

```
numbers = [2, 5, 1, 4]
n = len(numbers)

for i in range(n):
    for j in range(i):
        print(i, j)
```

Time complexity: ______________________

10) Coefficients don’t matter in Big-O

A program does about **3n + 100** operations.

Time complexity: ______________________

Part C — `break` and worst case

11) Search with `break`

```
numbers = [6, 1, 8, 9, 2]
target = 9

for x in numbers:
    if x == target:
        print("found")
        break
```

Fill in the blanks:

- Best case time complexity: ____________________
- Worst case time complexity: ____________________

12) Two-pointer palindrome check (concept review)

Assume `s` is a string of length `n`.

```
s = "ABCDCBA"
left = 0
right = len(s) - 1

while left < right:
    if s[left] != s[right]:
        print("not palindrome")
        break
    left = left + 1
    right = right - 1
```

Worst case time complexity: ____________________

Part D — Counting steps (small n)

13) How many prints? (n = 4)

```
numbers = [2, 5, 1, 4]

for x in numbers:
    print(x)
```

Number of `print` calls: ____________________

---

14) How many prints? (n = 3, nested loops)

```
n = 3

for i in range(n):
    for j in range(n):
        print(i, j)
```

Number of `print` calls: ____________________

Part E — Challenge

15) Unique pairs only (i < j)

```
numbers = [2, 5, 1, 4]
n = len(numbers)

for i in range(n):
    for j in range(i + 1, n):
        print(numbers[i], numbers[j])
```

1. Time complexity: ____________________
2. If `n = 4`, how many pairs are printed? ____________________