Quiz 13–16: While-Loops

Name: ____________________ Date: ____________________

Instructions

- Answer in the blanks.
- For “write code” questions, write valid Python code (no functions needed).
- For “what does it print” questions, write the **exact** output (line by line).
- You may assume all inputs are valid (unless the question says otherwise).

Part A — While-loop basics

1) While-loop facts (True/False)

Write **True** or **False** for each statement.

1. A `while` loop keeps running while its condition is `True`. __________
2. If a loop variable never changes, the `while` loop might become **an** infinite loop. __________
3. A `break` statement stops the loop immediately. __________

2) Count down (what does it print?)

```
x = 5
while x > 0:
    print(x)
    x = x - 2
print("done")
```

Output:

3) Fix the bug (infinite loop)

The code is supposed to print `3`, `2`, `1`, then stop.
There is a bug that can cause an infinite loop.

```
x = 3
while x >= 1:
    print(x)
    x = x + 1    # BUG: fix this line
```

Corrected line:

```
x = ______________________
```

4) First number bigger than 10 (what does it print?)

```
nums = [2, 5, 8, 11, 3]

i = 0
while i < len(nums):
    if nums[i] > 10:
        print("found:", nums[i])
        break
    print("not yet:", nums[i])
    i = i + 1
```

Output:

```

```

5) Count jumps by 4 (write code)

Write code that counts how many times you can add `4` starting from `start` before you would pass `limit`.

Example: if `start = 3` and `limit = 20`, you do: `3 → 7 → 11 → 15 → 19 → 23` (stop because the next value would pass 20).
So the answer is `5` jumps.

Complete the code:

```
start = 3
limit = 20

count = 0

while True:
    if ___________________________:
        break
    start = _______________________
    count = _______________________

print(count)
```

Part B — Digits (using `//` and `%`)

6) Count digits (what does it print?)

```
n = 2026
count = 0

while n != 0:
    count = count + 1
    n = n // 10

print(count)
```

Output:

```

```

7) Last digit + remove last digit (fill in the blanks)

Fill in the blanks so the code prints the digits of `n` from **right to left**.

```
n = 735

while n > 0:
    last = n % ______
    print(last)
    n = n // ______
```

Output:

```

```

8) Sum of digits (fill in the blanks)

Fill in the blanks so the code prints the sum of the digits of `n`.

```
n = 409
total = 0

while n > 0:
    digit = n % ______
    total = total + __________
    n = n // ______

print(total)
```

Expected output:

```
13
```

9) Collatz game (what does it print?)

Rules:

- If `n` is even, do `n = n // 2`
- If `n` is odd, do `n = 3*n + 1`

```
n = 7

while n != 1:
    print(n)
    if n % 2 == 0:
        n = n // 2
    else:
        n = 3*n + 1

print(1)
```

Output:

10) Factors using `while` (write code)

Write code to print all factors of `n` (numbers that divide `n` with remainder 0), one per line.

```
n = 12

i = 1

# complete the code:

while _____________________:
    if ____________________:
        print(i)
    i = ___________________
```

Part C — Palindrome with two pointers (strings)

11) Palindrome facts (True/False)

Write **True** or **False** for each statement.

1. A palindrome reads the same forward and backward. __________
2. With two pointers, `left` starts at the beginning (front) and `right` starts at the end (back). __________
3. If you find one mismatched pair of characters, you can stop early. __________

12) Trace the pointers (fill in)

Given:

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

Fill in:

- First comparison: `s[left]` is ________ and `s[right]` is ________
- After **one** successful match, `left` becomes ________ and `right` becomes ________
- Second comparison: `s[left]` is ________ and `s[right]` is ________

---

13) Read code and reason (what does it print?)

```
pal_word = "level"
not_pal_word = "hello"

# Check pal_word
s = pal_word
left = 0
right = len(s) - 1
ok = True
while left < right:
    if s[left] != s[right]:
        ok = False
        break
    left = left + 1
    right = right - 1
print(ok)

# Check not_pal_word
s = not_pal_word
left = 0
right = len(s) - 1
ok = True
while left < right:
    if s[left] != s[right]:
        ok = False
        break
    left = left + 1
    right = right - 1
print(ok)
```

Output:

14) Palindrome checker (write code)

Write code that prints `"YES"` if `s` is a palindrome, else prints `"NO"`.

```
s = "madam"

# complete the code:

left = ___________________
right = __________________
palindrome = _____________  # True or False?

while ____________________:
    if ___________________:
        palindrome = ______________
        __________________  # stop the loop

    left = _______________
    right = ______________

if palindrome:
    print(________________)
else:
    print(________________)
```

Part D — Swap and reverse

15) Swap two numbers (fill in the blanks)

Fill in the blanks so that `a` and `b` are swapped.

```
a = 6
b = 7

temp = ________
a = ________
b = ________

print(a, b)
```

Expected output:

```
7 6
```

---

16) Reverse a list in-place (what does it print?)

```
nums = [1, 2, 3, 4, 5]

left = 0
right = len(nums) - 1

while left < right:
    temp = nums[left]
    nums[left] = nums[right]
    nums[right] = temp
    left = left + 1
    right = right - 1

print(nums)
```

Output:

```

```

---

17) Fix the reverse bug (one line)

The code below is almost correct, but it has one bug in the pointer update.

```
nums = [10, 20, 30, 40]

left = 0
right = len(nums) - 1

while left < right:
    temp = nums[left]
    nums[left] = nums[right]
    nums[right] = temp

    left = left + 1
    right = right + 1   # BUG: fix this line

print(nums)
```

Corrected line:

```
right = ______________________
```

Part E — Move zeros to the end

18) Fill in the blanks: move zeros in-place

Fill in the blanks so the code moves all zeros to the end (stable order).

```
numbers = [0, 1, 0, 3, 0, 2]

write = _______
read = _______

# Step 1: copy nonzeros forward
while read < len(numbers):
    if numbers[read] != 0:
        numbers[_______] = numbers[_______]
        write = __________________________
    read = _______________________________

# Step 2: fill zeros from write to the end
k = write
while __________________________:
    numbers[k] = ______________________
    k = ______________________

print(numbers)
```

Expected output:

```
[1, 3, 2, 0, 0, 0]
```