

# Quiz 01 Practice

COMP 110: Introduction to Programming  
Fall 2025

Wednesday, September 17, 2025

**Question 1: Respond** to the following questions. If a line of code is unreachable, write "N/A."

Consider the following code listing:

```
1 def get_gift(budget: int) -> str:
2     """Returns a gift idea based on a budget."""
3     if budget > 0:
4         return "A gift!"
5     else:
6         if budget >= 110:
7             return "Nice dinner"
8         else:
9             return "Unreachable?"
```

1.1. Write a function call expression to the `get_gift` function that evaluates to "Unreachable?"

1.2. Write a function call expression to the `get_gift` function that evaluates to "Nice dinner"

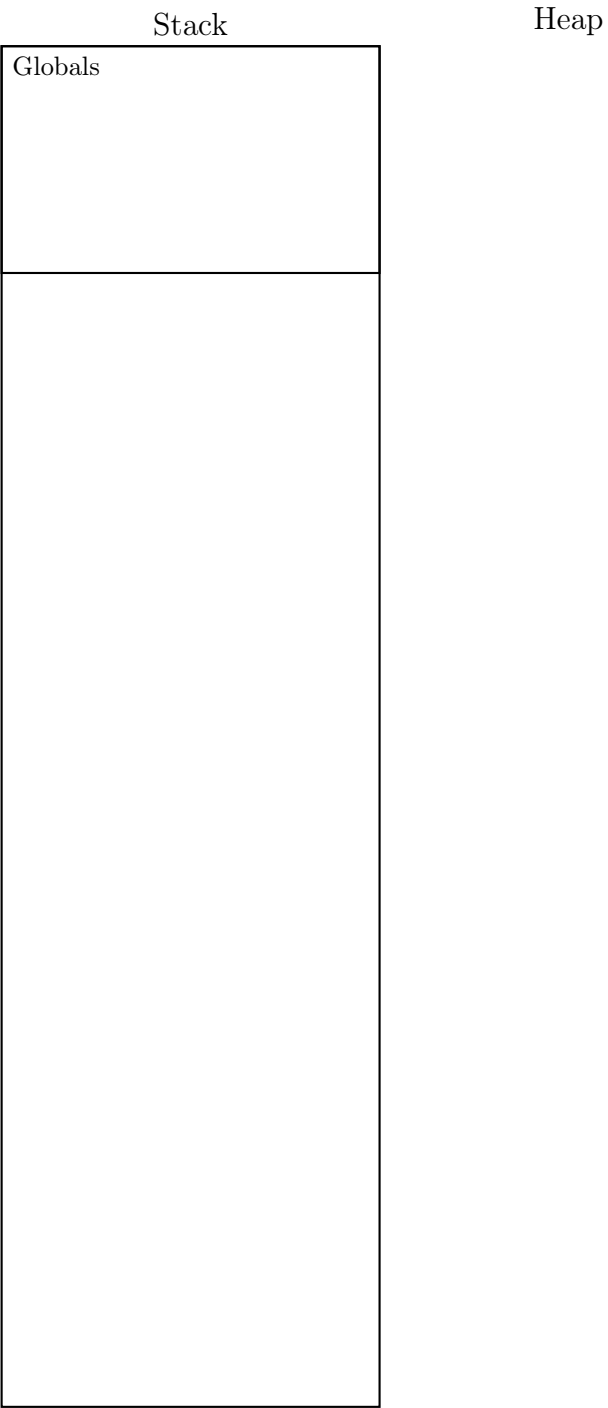
1.3. Write a function call expression to the `get_gift` function that evaluates to "A gift!"

1.4. Rewrite lines 3-9 of the code listing to eliminate unreachable code (if any) and the nested if-else statement. Your code should behave exactly as lines 3-9 of this code listing do; for any value of budget, your code should return exactly what lines 3-9 of the listing would return.

**Question 2: Memory Diagram** Trace a memory diagram of the following code listing.

```
1 def jump(word: str) -> str:
2     i: int = len(word) - 1
3     while i >= 0:
4         if i % 2 == 1:
5             print(word[i])
6         else:
7             print(i)
8             i = i - 1
9     return word[0]
10
11
12 jump(word="play")
```

Output



**Question 3: CHALLENGE: Recursive Function Definition Writing** Write a recursive function definition that returns the sum of all positive, even integers less than or equal to a given `int`. Your function definition should meet the following expectations:

- The function should be named `summit`, have one `int` parameter named `n`, and return an `int`.
- If `n` is negative, the function should return `-1`.
- If `n` is positive, the function should return the sum of all positive, even integers less than or equal to `n`.
- Explicitly type your parameters and return types.
- Label your base case(s) and recursive case(s).

The following REPL examples demonstrate the expected functionality of your `summit` function:

```
1 >>> summit(-2)
2 -1
3 >>> summit(1)
4 0
5 >>> summit(2)
6 2
7 >>> summit(3)
8 2
```

```
1 >>> summit(4)
2 6
3 >>> summit(5)
4 6
5 >>> summit(6)
6 12
7 >>> summit(12)
8 42
```

3.1. Write your function definition here:

*This page intentionally left blank. Do not remove from quiz packet.*