JamCoders 2025
Lecturers: Alex Krentsel, Timnit Gebru, Jelani Nelson, Orr Paradise

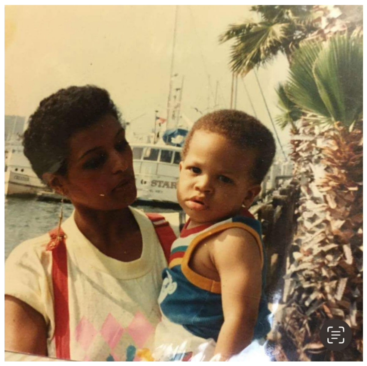
Exam, Week 3
July 18, 2025

Exam, Week 3

Name: ______________________________

Directions

- **This exam contains 18 questions (+1 optional question) and will last 90 minutes.**
- Use your time wisely. If you are having too much trouble on a question, skip it and return to it later. **Avoid getting stuck.**
- In the answer options, the ↵ symbol indicates a new line. The ↵ symbol will only be used to separate lines of output and will not appear at the end of the final line.
- If a question asks what is printed and nothing is printed, leave the line blank.
- For questions with *circular bubbles*, you should select *exactly one* choice.
  - ○ You must choose either this option
  - ○ Or this one, but not both!
- For questions with *square checkboxes*, you may select *multiple* choices.
  - ☐ You could select this choice.
  - ☐ You could select this one too!



Staff use only.

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Total
								(bonus)	

1. What will be printed after running the following code? Write your answers on the dashed lines.

```
a = 3
b = 4
c = 5
a = a + b - c
print(a) # Printed: _____________________________________
c = a * c + 5
print(c) # Printed: _____________________________________
a = c // a
print(a) # Printed: _____________________________________
```

2. What is $\log_7(49)$?

○ 1.73

○ 2

○ 7

○ None of the above

3. What will be printed after running the following code? Write your answers on the dashed lines.

```
A = [4, 0, 8]
B = [4, 1, 2]
L = A + B
print(L[5]) # Printed: _____________________________________
L = A + L
print(L[5]) # Printed: _____________________________________
```

4. For which value(s) of a and b will the following code print `Jelani` ⏎ `Nelson`?

```
if a:
    print('Timnit')
    if b:
        print('Gebru')
else:
    print('Jelani')
    if a or b:
        print('Nelson')
```

Fill in the boxes next to all answers that print `Jelani` ⏎ `Nelson`.

- ☐ `a = True, b = True`
- ☐ `a = True, b = False`
- ☐ `a = False, b = True`
- ☐ `a = False, b = False`

5. Binary search takes as input a sorted list `L` and an `element` and returns the index of the first occurrence of `element` in `L`, or `-1` if `element` is not found. Complete the following code implementing binary search.

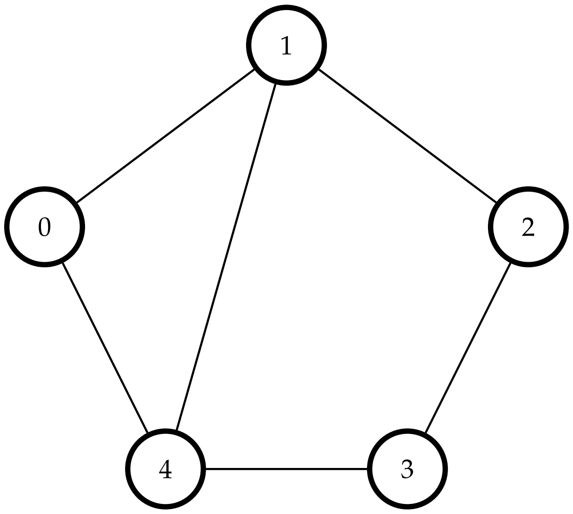
```
def binary_search(L, element):
    start = 0
    stop  = ________________________________________

    while __________________________________________:
        mid = (start + stop)//2
        if L[mid] < element:

            ______________________ = __________________________________
        elif L[mid] > element:

            ______________________ = __________________________________
        else:
            return mid
    return -1
```

6. Consider the following graph.



a) Mark all the neighbors of the vertex 1.

☐ 0
☐ 1
☐ 2
☐ 3
☐ 4

b) Complete the following code so that after it is run, the variable `G` stores the adjacency list of the graph.

```
G = ________________________________

    ________________________________

    ________________________________

    ________________________________

    ________________________________

    ________________________________

    ________________________________
```

7. Joy LOVES eggs. She has been keeping track of how many eggs she eats each day. Any day where Joy eats at least 5 eggs is considered a good day. Complete the following recursive function that takes in a list of integers denoting how many eggs Joy has eaten in each of the last few days, and returns how many of those days were good days.

Example: `num_good_days([1, 5, 24, 3, 10, 9, 0])` should return 4.

```
def num_good_days(L):
    if len(L) == 0:

        return ________________________________________

    if ________________________________________________:

        return num_good_days(L[1:]) + ________________________________
    else:
        return num_good_days(L[1:])
```

8. Consider the following three functions.

$$f(n) = \log_2(n) \times \log_2(n) \times n$$
$$g(n) = n^2(n^2 + 1)$$
$$h(n) = n^3$$

Rank these functions in order of asymptotic growth rate by filling in the blanks using f, g, and h. Use each exactly once:

As n gets large, ______(n) grows larger than ______(n), which grows larger than ______(n).

9. What will be printed after running the following code?

```
def frank_aura_points(n):
    if n <= 1:
        return 1
    return frank_aura_points(n - 1) + frank_aura_points(n - 2)

print(frank_aura_points(4))
```

- ○ 1
- ○ 3
- ○ 5
- ○ 8

10. What will be printed after running the following code? Write your answers on the dashed lines.

If nothing gets printed, write 'nothing' in the blank.

```
def print_between(start, stop):
    i = start
    while i < stop:
        print(i, end=' ')
        i += 2

print_between(0, 4)   # Printed: ____________________________
print()
print_between(5, 13)  # Printed: ____________________________
print()
print_between(8, 5)   # Printed: ____________________________
```

11. Complete the following `merge` function that takes in two sorted lists `L1`, `L2` and returns a sorted list containing all elements from both `L1` and `L2`.

```
def merge(L1, L2):
    out = []
    i = 0
    j = 0
    while i < len(L1) and j < len(L2):
        if L1[i] < L2[j]:

            _______________________________

            _______________________________
        else:
            _______________________________

            _______________________________
    out += L1[i:]
    out += L2[j:]
    return out
```

12. What is the running time of the following code in terms of `n`?

```
x = 0
for i in range(n):
    for j in range(n):
        for k in range(2**10):
            x += 2
```

- ○ $\mathcal{O}(2^n)$
- ○ $\mathcal{O}(n^2)$
- ○ $\mathcal{O}(n^2 2^n)$
- ○ $\mathcal{O}(n^3)$

13. What will be printed after running the following code?

```
def mystery(L):
    total = 0
    for x in L:
        total = x
    return total

print(mystery([2, 0, 2, 5]))
```

- ○ 0
- ○ 2
- ○ 5
- ○ 9

14. What will be printed after running the following code? Read the code carefully! Scratch work may be written in the column on the left. In the column on the right, include **only** the printed values.

If nothing gets printed, write 'nothing' in the blank.

```python
def piyush(x):
    print('Piyush')
    return x

def manolis():
    print('Manolis')
    return zaria()

def zaria():
    return 'Zaria'
```

a)

```python
piyush('hi')
```

b)

```python
print(piyush('hello'))
```

c)

```python
print(piyush(manolis()))
```

15. What does the following code print?

```
def mystery(lst):
    count = 0
    for i in range(len(lst)):
        for j in range(i + 1, len(lst)):
            if (lst[i] * lst[j]) % 2 == 0:
                count += 1
    return count

print(mystery([1, 2, 3, 4]))
```

○ 2

○ 4

○ 5

○ None of the above

16. What will be printed after running the following code?

```
def secret(L):
    if len(L) == 0:
        return
    secret(L[1:])
    print(L[0])
secret(['tek', 'it', 'easy'])
```

○ tek ↵ it ↵ easy

○ easy ↵ it ↵ tek

○ An error occurs, and nothing is printed.

○ tek ↵ tek ↵ tek

17. Write a function is_strictly_increasing which takes a list of integers lst and returns True if the list is sorted in strictly increasing order (each number is greater than the one before it), and False otherwise.

You cannot use the sorted or sort built-in function.

Example: is_strictly_increasing([1, 2, 3]) should return True.

Example: is_strictly_increasing([1, 2, 2, 3]) should return False.

Example: is_strictly_increasing([5, 4, 3]) should return False.

```
def is_strictly_increasing(lst):
    """
    Args: lst (list of int)
    Returns (bool): True if lst is strictly increasing, False otherwise.
    """
```

18. Write a function `filter_squares` that takes a list of integers **less than 50** `lst` and returns a new list containing all the numbers from `lst` that are **perfect squares**.

A perfect square is a number that has an integer square root (like 1, 4, 9, 16, . . .).

Example: `filter_squares([1, 2, 3, 4, 5, 25, 8, 9, 15])` should return `[1, 4, 25, 9]`.

Example: `filter_squares([7, 10])` should return `[]`.

Hint: note that there are not that many perfect squares up to 50!

```python
def filter_squares(lst):
    """
    Args: lst (list of int): The list of integers to check.
    Returns: list of int: Perfect squares.
    """
```

19. Bonus Problem: Below are four Python code snippets. Match each one to its most precise Big-O time complexity. Your answer should be as tight as possible—that is, choose the smallest class among those seen in lecture that accurately describes the code's worst-case runtime.

```python
# Snippet A
for i in range(n):
    for j in range(i):
        print(i + j)

# Snippet B
for i in range(n):
    x = i
    while x > 0:
        x = x // 2
        print(x)

# Snippet C
i = n
while i > 1:
    for j in range(i):
        print(i, j)
    i = i // 2

# Snippet D
for i in range(1, n):
    for j in range(1, n, i):
        print(i, j)
```

Snippet	Time Complexity
A	__________
B	__________
C	__________
D	__________