Exam, Week 3

Name:

Name of student to the left:

Name of student to the right:

Directions

- **This exam contains 19 questions and will last 90 minutes.**
- There are two long answer questions, worth 15 points each. The other 17 questions have you select from multiple choices or have you write a short answer, and are worth 5 points each. The maximum number of points is 115.
- 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

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

```
x = 2
y = 5
z = 4
x = x + y + z
print(x) # Printed: ________________________
z = x * y - x * z
print(z) # Printed: ________________________
x = x // z
print(x) # Printed: ________________________
```

2. What is $\log_3(81)$?

- ○ 4
- ○ 6.34
- ○ 9
- ○ 27

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

```
A = [2, 1, 0, 7]
B = [2, 0, 2, 3]
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 `Timnit ↵ Gebru`?

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

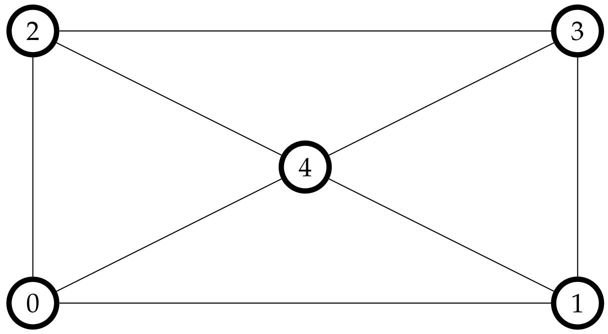
Fill in the boxes next to all answers that print `Timnit ↵ Gebru`.

- ☐ `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  # element not found in L
```

For the next two questions, consider the following graph.



6. Mark all the neighbors of the vertex 3.

- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4

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

```
G = _________________________________
    _________________________________
    _________________________________
    _________________________________
    _________________________________
    _________________________________
    _________________________________
    _________________________________
    _________________________________
```

8. Emaan has been keeping track of how many cockroaches he squishes each day. Any day where Emaan squishes at least 3 cockroaches is considered a good day. Complete the following recursive function that takes in a list of integers denoting how many cockroaches Emaan has squished in each of the last few days, and returns how many of those days were good days.

Example: `num_good_days([1, 5, 2, 7, 10, 0])` should return 3.

```
def num_good_days(L):
    if len(L) == 0:
        return ________________________________
    if ________________________________:
        return num_good_days(L[1:]) + ________________________________
    else:
        return num_good_days(L[1:])
```

9. Consider the following three functions.

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

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).

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

```
def ian_numbers(n):
    if n <= 0:
        return 2
    return ian_numbers(n - 1) + ian_numbers(n - 3)

print(ian_numbers(5))
```

- ○ 2
- ○ 5
- ○ 9
- ○ 18

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

```
def print_between(start, stop):
    i = start
    while i < stop:
        print(i)
        i += 1

print_between(5, 8)  # Printed: ____________________
print_between(8, 5)  # Printed: ____________________
```

12. 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
```

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

```
x = 1000
for i in range(n):
    for k in range(n):
        for j in range(5):
            x //= 2
```

- ○ $O(\log(n))$
- ○ $O(n^2)$
- ○ $O(n^2 \log(n))$
- ○ $O(n^3)$

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

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

print(mystery([1, 5, 2, 4]))
```

- ○ 4
- ○ 12
- ○ 1
- ○ 1524

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

```
def jon(x):
    print(x)
    return 'jon'

def ath(y):
    print('ath')
    return y

def an():
    return 'an'

jon('li')                 # Printed: ____________________
print(jon('li'))          # Printed: ____________________
print(jon(ath(an())))     # Printed: ____________________
```

16. What does the following code print?

```
def mystery(L):
    out = 0
    for i in range(len(L)):
        for j in range(len(L)):
            if L[i] + L[j] == 5:
                out += 1
    return out

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

- ○ 2
- ○ 4
- ○ 10
- ○ 30

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

```
def mystery(L):
    if len(L) == 1:
        return
    print(L[0])
    mystery(L[1])
mystery(['Big', 'up', 'yuh', 'self'])
```

- ○ Big ↵ u
- ○ An error occurs, and nothing is printed.
- ○ Big ↵ Big ↵ Big ↵ Big ↵ Big ↵ Big ↵ Big ↵ Big . . . (Infinitely printing Big ↵)
- ○ Big ↵ up ↵ yuh ↵ self

18. Write a function `is_sorted` which takes a list of integers `lst` and returns `True` if the list is sorted in non-decreasing order and `False` otherwise.

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

```
For example:
Arguments: lst = [1, 2, 3]
Returns: True
Arguments: lst = [3, 2, 1]
Returns: False
```

```python
def is_sorted(lst):
    """
    Args: lst (list of int)
    Returns (boolean): A boolean indicating if lst is sorted.
    """
```

19. Write a function `special_numbers` which takes a list of integers `lst` and returns a list of all the integers in `lst` that are divisible by 3 or 5, but not divisible by both 3 and 5.

```
For example:
Arguments: lst = [1, 2, 3]
Returns: [3]
Arguments: lst = [1, 2, 3, 5, 25, 15, 10, 35, 30, 27]
Returns: [3, 5, 25, 10, 35, 27]
```

```python
def special_numbers(lst):
    """
    Args: lst (list of int)
    Returns (list of int): The special numbers in lst.
    """
```