# Exam, Week 4

## Name:

## Name of student to the left:

## Name of student to the right:

## Name of Big Sib:

### Directions

- **This exam contains 20 questions and will last 100 minutes.**
- There are two long answer questions, worth 15 points each. The other 18 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 120.
- 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 answers, 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 | Q20 | | Total |
| | | | | | | | | | | |

**1.** For which value(s) of `a` and `b` will the following code print `Ian`?

```
if a or b:
    if a:
        print("Ian")
    else:
        print("Kerene")
else:
    if b:
        print("Jon")
    else:
        print("Jonathan")
```

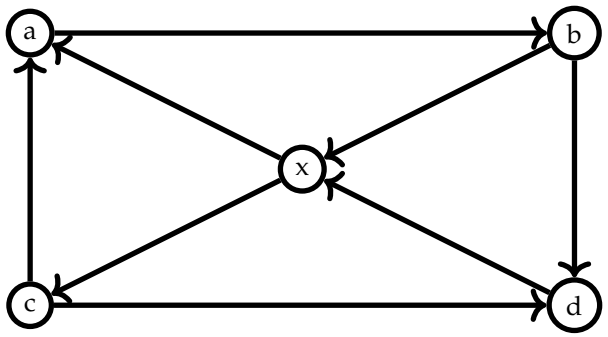
Fill in the boxes next to all answers that print `Ian`.

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

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

```
a = "hello"
b = "goodbye"
a = a + a
print(a) # Printed: _________________________
b = b[4:]
print(b) # Printed: _________________________
a = a + b
print(a) # Printed: _________________________
```

**3.** For the next question, consider the following directed graph.




Which of the following are paths in the graph?

- ☐ $a \to b \to d$
- ☐ $a \to b \to c \to d$
- ☐ $a \to b \to x \to c \to d$
- ☐ $a \to b \to x \to d$
- ☐ $a \to c \to d$

**4.** Complete the following code that finds the minimum element in a list.

```
def find_min(L):
    min_so_far = L[0]
    for element in L:
        if ________________________________________:
            min_so_far = ________________________________________
    return ________________________________________
```

**5.** Emaan used to consider a day to be good if he squished more than `x` cockroaches. He used the following code to count how many good days he's had.

```
def num_good_days(L, x):
    out = 0
    for element in L:
        if element > x:
            out += 1
    return out
```

Now Emaan is more merciful—he does not want to squish *too many* cockroaches. A good day is when he squishes more than `x` but less than `y` cockroaches. Write a function that returns how many good days are in a list `L` given these arguments.

```
def num_good_days(L, ____________________, ____________________):
    out = 0
    for element in L:
        if ________________________________________________:
            out += 1
    return out
```

**6.** Zaria has invented a new number sequence. The `n`'th number in the sequence is called $z_n$, defined as follows:

$$z_0 = 19$$

$$z_1 = 17$$

$$z_n = 21 \times z_{n-1} + 6 \times z_{n-2} + 14$$

For example, the first few numbers in the sequence are $z_0 = 19, z_1 = 17, z_2 = 485$.

She has started to implement a function that calculates the `n`'th Zaria number, but didn't have time to write the base case. She asks you to finish the code written below.

```
def zaria_number(n):
    if _________________________________:
        _________________________________
    if _________________________________:
        _________________________________

    return 21 * zaria_number(n - 1) + 6 * zaria_number(n - 2) + 14
```

**7.** The implementation of `binary_search(L, item)` from lab is provided below.

```
def binary_search(L, item):
    left = 0
    right = len(L) - 1
    while left <= right:
        mid = (left + right) // 2
        if L[mid] < item:
            left = mid + 1
        elif L[mid] > item:
            right = mid - 1
        else:
            return mid
    return -1
```

What does `binary_search([2, 8, 5, 9, 10], 8)` return? Note that the input list is **not** sorted.

- ○ -1
- ○ 1
- ○ 2
- ○ An error occurs and nothing is returned.

**8.** What will be printed after running the following code? If there would be an error, write `error`. Write your answers on the dashed lines.

```
d = {
    'ice': 'cube',
    2: 'pac',
    'another': ['one', 'bites', 'the', 'dust']
}

print(d['ice'])         # prints _____________________________________
print(d['cube'])        # prints _____________________________________
print(d[2])             # prints _____________________________________
print(d[1])             # prints _____________________________________
print(d['another'][1]) # prints _____________________________________
```

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

```
def mystery(s):
    if s == '':
        return ''
    if s[0] == 'o':
        return 'a' + mystery(s[1:])
    return s[0] + mystery(s[1:])

print(mystery('abc'))      # prints _____________________________________
print(mystery('hola!'))    # prints _____________________________________
print(mystery('otomatik')) # prints _____________________________________
```

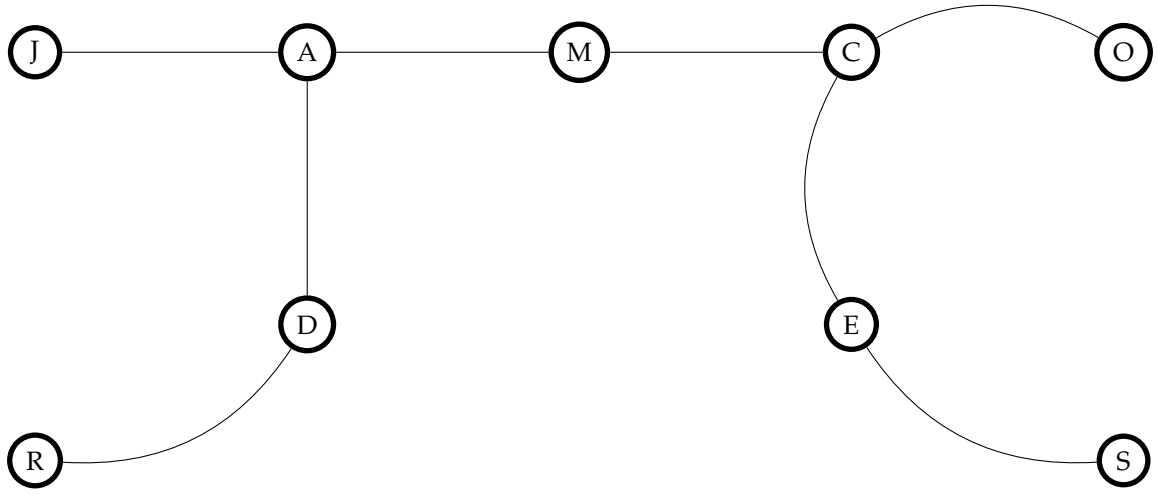
**10.** What will be printed after running the following code? Write your answers on the dashed lines.

```
def foo(x, y):
    print(x)
    return y

def bar(a, b):
    print(a + b)

foo(1, 2)         # Printed: _________________________
print(foo(3, 4))  # Printed: _________________________
bar(foo(1, 2), 3) # Printed: _________________________
```

**11.** Recall that Depth First-Search (DFS) and Breadth First-Search (BFS) are algorithms that traverse ("visit") nodes in a graph. Consider the following undirected graph.




In what order are nodes traversed ("visited") by DFS run on this graph starting from node $J$? There may be multiple correct answers, write just one.

In what order are nodes traversed ("visited") by BFS run on this graph starting from node $J$? There may be multiple correct answers, write just one.

**12.** Consider the following three functions.

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

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 ___$(n)$.

**13.** The following code defines two functions, `f` and `g`.

```
def f():
    for x in range(2 ** 10):
        print('work')

def g(n):
    for i in range(n):
        for j in range(n // 2):
            f()
```

The asymptotic ("Big O") running time of `f` in terms of `n` is

- ○ $O(1)$
- ○ $O(\log(n))$
- ○ $O(n^{10})$
- ○ $O(2^n)$

The asymptotic ("Big O") running time of `g` in terms of `n` is

- ○ $O(n2^n)$
- ○ $O(n^2)$
- ○ $O(n^{12})$
- ○ $O(n\log(n))$

**14.** An *inversion* in a list $L$ is a pair of indices $[i, j]$ such that $i < j$ and $L[i] > L[j]$.

For example, $[3, 1, 2]$ has two inversions: $[0, 1]$ and $[0, 2]$.

Complete the following function `count_inversions(L)` that takes in a list of integers `L` and returns the number of inversions in the list.

```
def count_inversions(L):
    out = 0
    for i in range(len(L)):
        for j in range(len(L)):
            if ______________________________________________:
                out += 1
    return out
```

**15.** What will be printed after running the following code?

```
def magic(lst):
    if len(lst) == 1:
        return lst[0]
    return lst[0] + lst[1] + magic(lst[1:])

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

- ○ 10
- ○ 15
- ○ 19
- ○ 20

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

```
def mystery(n, output):
    if n > 0:
        output.append(n)
        mystery(n - 1, output)
        mystery(n - 2, output)
    return output

print(mystery(1, []))        # prints _____________________________
print(mystery(1, ["hello"])) # prints _____________________________
print(mystery(2, []))        # prints _____________________________
print(mystery(3, []))        # prints _____________________________
print(mystery(4, []))        # prints _____________________________
```

**17.** Complete the function `count_a(L)`. The function takes a list of strings `L` as an argument and returns the number of times the character `'a'` appears in all strings in the list.

For example,

`count_a(['san francisco', 'california'])` should return 4

`count_a(['pawat', 'alok', 'zaria', 'emaan'])` should return 7

`count_a(['jon', 'kerene'])` should return 0

```
def count_a(L):
    if L == []:
        return ______________________
    counter = 0
    first_word = ______________________
    for character in first_word:
        if character == 'a':
            ______________________
    return ______________________ + count_a(______________________)
```

**18.** Micah has a list of integers and wants to find out if he can select a few of them so they sum up to some target number (each element can be selected once, or not at all).

For example, if the list is `[2, 1, 2, 6]` and Micah's target is 5, then Micah can select 1, 2 and 2 to sum up to 5. But if the given list is `[1, 3, 3]`, then there is no way to select items from the list that sum to 5.

Write a function `can_sum(L, target)` that takes in a list of integers `L` and the target number `target`. The function should return `True` if there is a way to select elements from `L` that sum to `target`, and `False` otherwise.

Complete the following code that solves this problem.

```
def can_sum(L, target):
    if target == ____________________:
        return True
    if ____________________:
        return False
    return can_sum(L[1:], ___________) or can_sum(L[1:], ___________)
```

**19.** Write a function `sum_more_than(L, k)` that takes as inputs a list of integers `L` and outputs the sum of all elements that are more than `k`.

For example, `sum_more_than([1, 4, 1, 5], 3)` returns `9`. `sum_more_than([2, -1, 3], 2)` returns `3`.

```
def sum_more_than(L, k):
    """
    Args:
        L (list[int]): List of integers.
        k (int): An integer.
    Returns (int): Sum of elements in L that are more than k.
    """
```

**20.** The mango tree in front of Rex has 0 mangoes on day 0. Every day, the number of mangoes on the tree increases by 8. At the end of every 3 days (i.e., day 0, 3, 6, ...), Pawat has a mango craving and picks half the mangoes from the tree, rounded down to the closest integer. Write a function `num_mangoes(n)` which determines the number of mangoes left on the tree at the end of day `n`. For example, `num_mangoes(4)` returns 20:

- Day 0: 0
- Day 1: 8
- Day 2: 16
- Day 3: 12, because there were 24 and Pawat picked half.
- Day 4: 20

```
def num_mangoes(n):
    """
    Args: n (int): The day we are interested in.
    Returns (int): The number of mangoes on the tree at the end of day n.
    """
```

## Bonus (0 points)

You are standing outside a building with 400 floors, each labelled in order from $1$ to $400$. You have two eggs. You know there is some floor $1 \leq x \leq 400$ such that dropping an egg from the $x$'th floor or lower will not break it, but dropping it from any higher floor will break it. You wish to find out what $x$ is.

As mentioned before, you have two eggs. If you drop one and it breaks, you may not re-use it. But if you drop one and it doesn't break, you may use it in future drops. You wish to find out what $x$ is using the smallest number of drops as possible.

One way to do this is to first drop an egg from floor $1$, then from floor $2$, and so on until it breaks. This uses at most $400$ drops and can be accomplished with just one egg. On the other end of the spectrum, if you had unlimited eggs (so you can break as many eggs as you like), you could binary search for $x$ and find it in at most $9$ drops.

Find a strategy that uses at most two eggs and as few drops as possible.