Exam, Week 4

Name: ______________________________

Directions

- **This exam contains 20 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.**
- If a question asks what is printed and nothing is printed, write ‘nothing‘ in the 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	Q21	Total
									(bonus)	

1. Consider the following code.

```
if a or b:
    if a:
        print("Sam")
    else:
        print("Bruno")
else:
    if b:
        print("Hanna")
    else:
        print("Frank")
```

Fill in the boxes next to all answers for which the code will print `Bruno`.

- ☐ `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 = "Kings"
b = "ton"
c = a + b * 3
print(c)        # Prints: ________________________________
b = c[1:4]
print(b)        # Prints: ________________________________
a = a + b
print(a)        # Prints: ________________________________
```

3. Complete the following code to return the maximum element in a non-empty list of integers.

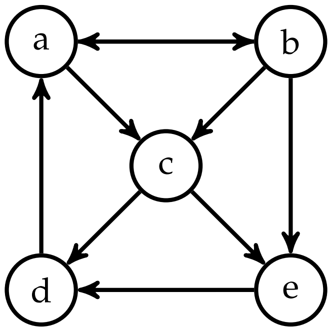
```
def find_max(L):
    max_so_far = L[0]
    for element in L:

        if __________________________________________:

            max_so_far = __________________________________________

    return __________________________________________
```

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



Which of the following are cycles in the graph?

☐ $a \to b \to d \to a$

☐ $a \to c \to d \to a$

☐ $a \to b \to e \to d \to a$

☐ $d \to a \to b$

☐ $d \to c \to e \to d$

5. Xavier invented a new number sequence. The n-th number in the sequence, z_n, is defined as follows:

$$\begin{aligned} z_0 &= 17 \\ z_1 &= 4 \\ z_n &= 12 \cdot z_{n-1} - 7 \cdot z_{n-2} + 4 \end{aligned}$$

For example, the first few numbers in the sequence are $z_0 = 17, z_1 = 4, z_2 = -67$.

He has started to implement a function that calculates the n-th Xavier number, but didn't have time to write the base case. Finish the code below.

```
def xavier_number(n):

    if ________________________________:

        ______________________________________________________

    if ________________________________:

        ______________________________________________________
    return 12 * xavier_number(n - 1) - 7 * xavier_number(n - 2) + 4
```

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

```
u = "you"
d = {
    "rubiks" : "cube",
    1 : "more time",
    "never" : ["going", 2, "give", u, "up"]
}

print(d["rubiks"])      # Prints: ________________________________

print(d["cube"])        # Prints: ________________________________

print(d[1])             # Prints: ________________________________

print(d[0])             # Prints: ________________________________

print(d["never"][3])    # Prints: ________________________________
```

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

```
def magic(lst):
    if len(lst) == 1:
        return 2 * lst[0]
    return lst[0] + lst[-1] + magic(lst[:-1])

print(magic([1, 3, 5, 7]))
```

- ○ 13
- ○ 19
- ○ 20
- ○ 37

8. Joy LOVES eggs. She used to consider any day on which she ate at least x eggs to be a good day. The function below counts how many good days she had, given a list L of integers representing the number of eggs she ate each day.

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

Now Joy is a bit more mindful of the chickens – she does not want to make them work too hard! A good day is when she eats **at least** x but **less than** y eggs. 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
```

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

```
def mystery(s):
    if len(s) == 0:
        return s
    if s[0] == s[0].upper():
        return s[0]*2 + mystery(s[1:])
    return s[0] + mystery(s[1:])

print(mystery('aBc'))      # Prints: _______________________________

print(mystery('hola'))     # Prints: _______________________________

print(mystery('PATTY'))    # Prints: _______________________________
```

10. Consider a directed graph represented by the following adjacency list:

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

Draw what this graph looks like.

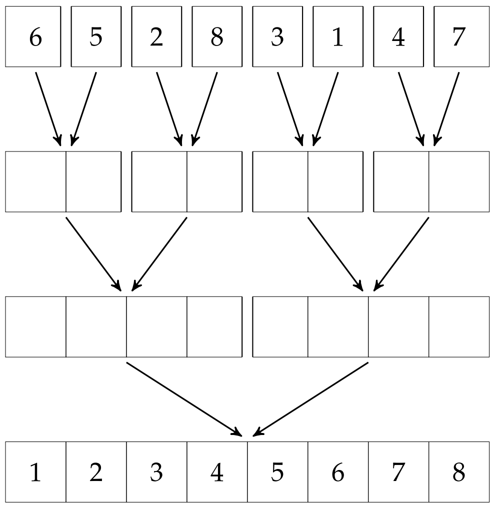
Write the adjacency matrix representation of the graph.

11. Consider the following unsorted list of numbers: `L = [6, 5, 2, 8, 3, 1, 4, 7]`. We want to sort `L` in increasing order. In the **left column**, show step-by-step how **selection sort** would sort the array. In the **right column**, show step-by-step how **merge sort** would sort the array. The initial and final states of the array are provided for you. If a number stays in the same column for the remainder of the sort, feel free to draw a vertical arrow all the way down instead of rewriting the number.

Selection Sort

6	5	2	8	3	1	4	7
1	2	3	4	5	6	7	8

Merge Sort



Now that the array is sorted, we can use `binary_search(L, item)` to find the position of an item in the list. We provide the implementation below; notice that each time `mid` is updated, its value is also printed.

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

What numbers will be printed by `binary_search([1, 2, 3, 4, 5, 6, 7, 8], 6.2)`?

12. 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 zaria(x):
    return "Manolis " + x

def piyush(x, y):
    print(x)
    return y

def manolis(y):
    print("Piyush")
    return zaria(y)
```

a)

```
print(piyush("Zaria", "Sam"))
```

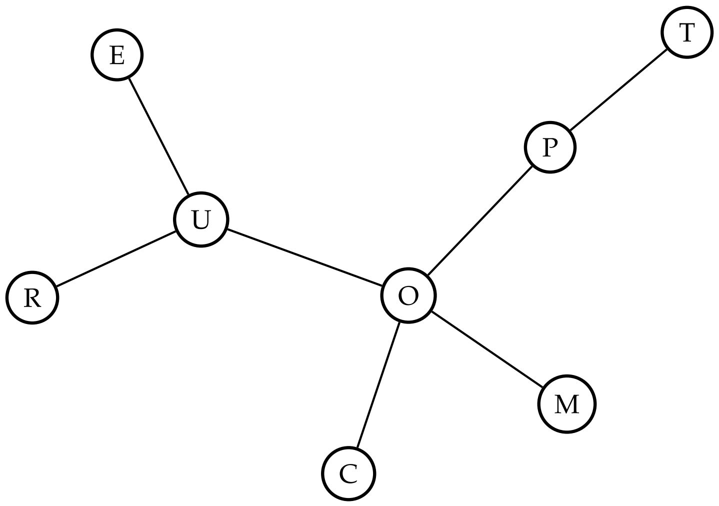
b)

```
print(piyush("Sam", zaria("Hanna")))
```

c)

```
print(manolis("Zaria"))
```

13. 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.



DFS: In what order are the nodes visited during a *depth-first search* on the graph, starting from node C? If there is a tie (i.e., multiple unvisited neighbors are available), break the tie alphabetically — for example, choose `A` before `B`.

BFS: In what order are the nodes visited during a *breadth-first search* on the graph, starting from node C? As with DFS, if multiple nodes are available to visit next, break ties alphabetically — for example, choose `A` before `B`.

Alphabet: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

14. Consider the following three functions.

$$
\begin{aligned}
f(n) &= 10n^{10} + 7n^7 + 4n^4 + n \\
g(n) &= \log n \times (\log n + \log n \times \log n) \\
h(n) &= n + n^2 + 2^n
\end{aligned}
$$

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).

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

```
def f(n):
    for i in range(n):
        for j in range(n):
            print('work')
def g(n):
    for i in range(n):
        f(i)
```

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

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

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

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

16. Adrianna is learning a new dance routine. The function `dance(n, moves)` builds a sequence of steps where each `"L"` represents a step to the left, and each `"R"` a step to the right. Whenever the function appends `"L" * n`, she steps left `n` times; and similarly, `"R" * n` means stepping right `n` times. What will be printed after running the following code? Write your answers on the dashed lines.

```
def dance(n, moves):
    if n > 0:
        moves.append("L" * n)
        dance(n - 1, moves)
        moves.append("R" * n)
        dance(n - 2, moves)
    return moves

print(dance(1, []))             # Prints: ________________________________

print(dance(1, ["adrianna"]))   # Prints: ________________________________

print(dance(3, []))             # Prints: ________________________________
```

17. Complete the function `func`, which takes as input a list of strings `lst`, and returns `True` if the list satisfies both of the following conditions:

- The first string is equal to the last, the second is equal to the second-to-last, and so on.
- If the list has an odd length, the middle string must start with the letter `"b"`.

Otherwise, the function returns `False`.

Examples:

- `func(["ab", "cd", "cd", "ab"])` returns `True`
- `func(["hi", "bye", "boat", "bye", "hi"])` returns `True` (middle string `"boat"` starts with `"b"`)
- `func(["hi", "apple", "hi"])` returns `False` (middle string `"apple"` doesn't start with `"b"`)
- `func(["hello", "world"])` returns `False`

```python
def func(lst):
    if len(lst) == 0:
        return True
    elif len(lst) == 1:

        return _______________ == _______________
    if lst[0] == lst[-1]:

        return ________________________________
    return False
```

18. Complete the function `count_doubles`. The function takes a list of strings `L` as an argument and returns the number of strings in `L` that have the same letter twice in a row.

Examples:

- `count_doubles(["count", "double", "letters", "soon"])` returns 2 (for `"letters"` and `"soon"`)
- `count_doubles(["big", "little", "eye"])` returns 1 (for `"little"`)
- `count_doubles(["mississippi"])` returns 1 (for `"mississippi"`)

```python
def count_doubles(L):
    if len(L) == 0:
        return 0

    count = count_doubles(________________)
    has_double = False
    for i in range(__________, __________):

        if _____________________________:
            has_double = True

    if _____________________________:
        count += 1
    return count
```

19. Write a function `filter_out(L, banned)` that takes as input two lists of integers, `L` and `banned`. The function should return a new list containing all elements of `L` that are *not* in `banned`, preserving their original order.

Examples:

- `filter_out([1, 2, 3, 4, 5], [2, 4])` returns `[1, 3, 5]`
- `filter_out([7, 8, 9, 8], [])` returns `[7, 8, 9, 8]`
- `filter_out([1, 2, 2, 3], [4, 2])` returns `[1, 3]`

20. Frank and Lydia visited the Chinese Garden at Hope Gardens, where there is a beautiful pond of water lilies. They liked it so much they keep coming back to harvest flowers.

At the start of day 0, there are 0 water lilies growing on the pond. Each day, 9 new lilies spring up. However, at the end of every 3 days (i.e., day 0, 3, 6, ...), Frank gathers half of all the flowers, rounded down to the closest integer. At the end of every 5 days (i.e., day 0, 5, 10, ...), Lydia takes 10 lilies. If Frank and Lydia come to the pond on the same day, first Lydia takes 10 lilies, and then Frank gets to keep half of what is left.

Write a function `num_lilies(n)` which determines the number of lilies floating in the pond at the end of day `n`. For example, `num_lilies(6)` returns 16:

- Day 0: 0
- Day 1: 9
- Day 2: 18
- Day 3: 14, because there were 27 and Frank took half (13).
- Day 4: 23
- Day 5: 22, because there were 32 and Lydia took 10
- Day 6: 16, because there were 31 and Frank took half (15)

21. Bonus Problem: Only attempt this problem if you're done with the exam!

Orr is scrambling to do his laundry. It costs `n` dollars to run the washing machine, but he's all out of coins.

Thankfully, you're ready to help. You have an infinite supply of certain types of coins. For example, you might have `c = [1, 5, 10, 20]`, meaning you can use as many \$1, \$5, \$10, and \$20 coins as you like. Write a function, `laundry_coins(n, c)`, that returns the minimum number of coins needed to make exactly `n` using the coin values listed in `c`. Assume that `n` is a positive integer, and that the list `c` always includes the coin `1`, so it is always possible to make exact change for any amount.

Examples:

- `laundry_coins(14, [1, 5, 10, 20])` returns 5 (\$10 + \$1 + \$1 + \$1 + \$1)
- `laundry_coins(406, [1, 10, 20])` returns 26 (twenty \$20 coins + six \$1 coins)
- `laundry_coins(25, [1, 5])` returns 5 (five \$5 coins)
- `laundry_coins(12, [1, 4, 5])` returns 3 (three \$4 coins)

You get an extra 5 bonus points if you use memoization (saving answers to subproblems), but this is not required to solve the problem!

Hint: Try solving small cases by hand first. Think about how you can try each coin and use recursion to find the best among all possible first choices.