Final Exam

Name:

Name of student to the left:

Name of student to the right:

Directions

- **This exam contains 18 questions and will last 90 minutes.**
- Two questions are long answer, while the other 16 questions have you select from multiple choices or have you write a short answer.
- 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.
- 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				Total

1. Select all statements below that are TRUE about recursion.

- ☐ Every recursive function must have a base case.
- ☐ Recursion can only be used to solve mathematical problems.
- ☐ Each recursive step breaks down the problem into a smaller, simpler sub-problem.
- ☐ Every recursive function must call itself.
- ☐ Recursion is really fun!

2. Consider the following code and fill in the blanks next to each print statement.

```
apple = 23
apple = apple // 2
print(apple)        # What is printed: _______
apple = apple / 2
print(apple)        # What is printed: _______
banana = 5
orange = 2
banana *= orange + 1
print(banana)       # What is printed: _______
orange = banana % 4
print(orange)       # What is printed: _______
```

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

```
plants = ["palm", "coconut", "orchid"]
numbers = [5, 0, 14, 37]
other_numbers = [2, 5]
print(plants[:2])           # Printed: ________________
numbers += other_numbers[1:]
print(numbers)              # Printed: ________________
print(other_numbers)        # Printed: ________________
other_numbers = other_numbers[:1]
print(other_numbers)        # Printed: ________________
print(plants[numbers[1]])   # Printed: ________________
```

4. Which of the following are valid comparisons of time complexity? Select all that apply. (Note: If a < b, this means a is faster than b.)

- ☐ O(1) < O(n)
- ☐ O(n) < O(log n)
- ☐ O(n) < O(n!)
- ☐ O(log n) < O(1)
- ☐ O(n^2) < O(n)

5. Consider the following code.

```
lst = [1, 2, 3, 4, 5]
a = 6
b = 3
word = "jamcoders"

print(lst[a // b])  # Line 1
print(lst[b // a])  # Line 2
print(a[b])         # Line 3
print(word[lst[b]]) # Line 4
print(lst[a + b])   # Line 5
```

Which of the following lines of code contain an invalid operation (meaning they will error when run)? Select all that apply.

- ☐ Line 1
- ☐ Line 2
- ☐ Line 3
- ☐ Line 4
- ☐ Line 5

6. `staff` is a dictionary containing the names of lecturers and teaching assistants. Fill in the blanks to accomplish the tasks below using **dictionary and list concepts**.

```
staff = {'B': ['Boaz'], 'T': ['Tarun', 'Timnit'],
'X': ['Xavier', 'Kimberli']}
```

Using **dictionary concepts**, print the list of teaching staff whose names begin with B.

_____

Using **dictionary and list concepts**, print "Timnit" (i.e., do not use `print("Timnit")` or similar to do this).

_____

Use list slicing to remove "Kimberli" from the dictionary.

_____

Create a new key-value pair in `staff` with the key `"K"` and value being a list containing the string `"Kimberli"`.

_____

7. Consider the following lines of code.

```
x = 14
if x % 2 == 0 and x < 10:
    print("Good")
else:
    print("Amazing")

y = 8
if y - 2 == 5 or y < 10:
    print("Day")
else:
    print("Night")
```

Which of the following are printed? Select all that apply.

- ☐ Good
- ☐ Amazing
- ☐ Day
- ☐ Night

8. Consider the following code.

```
if a and b:
    print("James")
if a or b:
    print("Zaria")
if not a:
    print("Liam")
if a and not b:
    print("Ecy")
if not a or not b:
    print("Reggie")
```

Which TA's names are printed when `a = True` and `b = False`? Select all that apply.

- ☐ James
- ☐ Zaria
- ☐ Liam
- ☐ Ecy
- ☐ Reggie

9. Consider the following lines of code. Fill in the blanks next to each line.

```
def beep(x):
   print("beep")

def boop(y):
   print(y)
   return "boop"

def buzz(z):
   return z

boop("jamcoders")              # What is printed: _________________
print(buzz(boop("coding")))    # What is printed: _________________
print(beep("python"))          # What is printed: _________________
```

10. Fill in the blanks below with the running time of each segment of code. Use big-O notation and express your answer in terms of the input size n. (You can assume that any line containing math operations and assignment of variables takes 1 unit of time.)

```
n = 2                    # Line 1
for i in range(n):       # Line 2
    n *= 2               # Line 3
```

The running time of line 1 is O(_____)

The running time of line 2 is O(_____)

The running time of line 3 is O(_____)

What is the total running time of this code?

- ○ O(1)
- ○ O(n^2)
- ○ O(n)
- ○ O(log n)

11. Which of the below steps are involved in recursive merge sort? Select all that apply.

- □ Swap the smallest element in the list with the first element in the list
- □ Split the list into 2 halves
- □ Check if the length of the list equals 1
- □ Combine 2 sorted lists into 1 sorted list
- □ Combine 2 unsorted lists into 1 sorted list

Consider the recursive function below **for the next two questions (12 and 13)**.

```
def mystery(n):
    if n == 1:
        return 1
    else:
        if n % 2 == 0:
            return mystery(n-1)
        else:
            return n + mystery(n-1)
```

12. What does `mystery(5)` return? _____

13. Which of the following best describes what the `mystery` function is doing?

- ○ Return the sum of numbers from 1 to n
- ○ Return the product of numbers from 1 to n
- ○ Return the sum of even numbers from 1 to n
- ○ Return the product of odd numbers from 1 to n
- ○ Return the sum of odd numbers from 1 to n

14. Anita and Xavier enjoy eating Welch's fruit gummies, and their favourite flavor is peach. Help them write a recursive function that returns the number of peach gummies in a list.

```
For example:
Argument: lst = ["orange", "berry", "peach", "apple", "peach"])
Returns: 2
```

```
def count_peach(lst):
    """
    Args: lst (list of str)
    Returns (int): the number of times "peach" occurs in lst
    """
    # Base case
    if _________________________________:
        return ______________________________
    else:
        # Recursive step
        if ______________________________:
            return ______________________________
        else:
            return ______________________________
```

15. Below is an implementation of binary search using a while loop. What does this function return when `lst = [1, 2, 4, 6, 7, 10]` and `target = 3`?

```
def binary_search(lst, target):
    """
    Finds target in a sorted list lst.
    Args:
        lst (list(int)):
            List of numbers to search through.
        target (int):
            Number we'd like to find.
    Returns: ?
    """
    left = 0
    right = len(lst) - 1

    while left <= right:
        mid = (left+right) // 2
        if lst[mid] == target:
            break
        if lst[mid] > target:
            right = mid - 1
        else:
            left = mid + 1
    return mid
```

- ○ -1
- ○ 1
- ○ 3
- ○ 4

16. The function `averages` has been partially implemented. It takes a nested list of integers `lst` and returns a list of averages, one for each of `lst`'s sublists. Fill in the blanks to complete this function.

```
For example:
Arguments: lst = [[0, 1, 2], [16, 14, 15], [10, 10, 7]]
Returns: [1.0, 15.0, 9.0]
```

```
def averages(lst):
    """
    Args: lst (nested list of int)
        List of sublists for which we'd like the averages.
    Returns (list of int):
        List containing the average of each sublist in lst.
    """
    result = []
    for ____________________________________:
        sum = 0
        for _________________________________:
            sum += ________________________
        avg = sum / len(________________________)
        result.append(_________________________)
    return result
```

17. Write a function `length_sum` that is given a list of strings `lst` and returns the sum of all string lengths in the list. This function should accomplish this using **recursion**.

```
For example:
Argument: lst = ["hello", "jamcoders", "!!"]
Returns: 16
Explanation: len("hello") = 5, len("jamcoders") = 9, and len("!!") = 2.
We return 5 + 9 + 2 = 16.
```

```
def length_sum(lst):
    """
    Args: lst (list of str)
    Returns (int): the sum of all string lengths in lst
    """
```

18. Write a function `reverse_string` that takes a string `str` and returns the reversed version of the string. This function should accomplish this using **while loops**. Do **not** use `reverse()` or string/list slicing to reverse the input.

```
For example:
Argument: str = "castleton"
Returns: "noteltsac"
Argument: str = "exceptional"
Returns: "lanoitpecxe"
```

```
def reverse_string(str):
    """
    Args: str (string to be reversed)
    Returns (string): the reversed version of str
    """
```