PID or Name: ______________________

DSC 40B - Sample Midterm 01

Note: This sample midterm is intended to give you an idea of the format of the exam, but it's not intended to be a comprehensive review of the material. Also, note that this sample exam is from a previous iteration of the course, and topics can change slightly from quarter to quarter depending on the instructor and how much was covered in lecture – you should expect Midterm 01 to cover the content from *this quarter*'s Lecture 01 through 08.

In addition to this sample exam, you should also study using the problems found at `https://dsc40b.com/practice`, as well as the labs and the homeworks.

Problem 1.

What is the time complexity of the following function? State your answer as a function of n using asymptotic notation in the simplest form possible. E.g., "$\Theta(n)$".

```
def foo(n):
    for i in range(n**3):
        for j in range(n):
            print(i + j)
        for j in range(n**2):
            print(i + j)
```

Problem 2.

What is the time complexity of the following function? State your answer as a function of n using asymptotic notation in the simplest form possible. E.g., "$\Theta(n)$".

```
def foo(n):
    for i in range(n):
        for j in range(i):
            for k in range(n):
                print(i + j + k)
```

Problem 3.

What is the time complexity of the following function? State your answer as a function of n using asymptotic notation in the simplest form possible. E.g., "$\Theta(n)$".

```
def foo(n):
    for i in range(200, n):
        for j in range(i, 2*i + n**2):
            print(i + j)
```

Problem 4.

What is the time complexity of the following function? State your answer as a function of n using asymptotic notation in the simplest form possible. E.g., "$\Theta(n)$".

```
import math

def foo(arr):
    """`arr` is an array with n elements."""
    n = len(arr)
    ix = 1
    s = 0

    while ix < n:
        s = s + arr[ix]
        ix = ix * 5 + 2

    return s
```

Problem 5.

The code below takes in an array of n numbers and checks whether there is a pair of numbers in the array which, when added together, equal the maximum element of the array.

What is the **best case** time complexity of this code as a function of n? State your answer using asymptotic notation.

```
def exists_pair_summing_to_max(arr):
    n = len(arr)
    maximum = max(arr)
    for i in range(n):
        for j in range(i + 1, n):
            if arr[i] + arr[j] == maximum:
                return True
    return False
```

Problem 6.

What is the **worst case** time complexity of the function in the last problem? State your answer using asymptotic notation.

Problem 7.

Consider again the problem of determining whether there exists a pair of numbers in an array which, when added together, equal the maximum number in the array. Additionally, **assume that the array is sorted**.

True or False: $\Theta(n)$ is a **tight** theoretical lower bound for this problem.

◯ True

◯ False

Problem 8. (2 points)

Suppose a and b are two numbers, with $a \leq b$. Consider the problem of counting the number of elements in an array which are between a and b; that is, the number of elements x such that $a \leq x \leq b$. You may assume for simplicity that both a and b are in the array, and there are no duplicates.

a) What is a **tight** theoretical lower bound for this problem, assuming that the array is **unsorted**? State your answer in asymptotic notation as a function of the number of elements in the array, n.

b) What is a **tight** theoretical lower bound for this problem, assuming that the array is **sorted**? State your answer in asymptotic notation as a function of the number of elements in the array, n.

Problem 9.

What is the **expected** time complexity of the following function? State your answer using asymptotic notation.

```
import random

def foo(n):
    x = random.randrange(n)

    if x < 8:
        for j in range(n**3):
            print(j)
    else:
        for j in range(n):
            print(j)
```

Problem 10.

What is the **expected** time complexity of the function below? State your answer using asymptotic notation. You may assume that `math.sqrt` and `math.log` take $\Theta(1)$ time. `math.log` computes the natural log.

```
import random
import math

def foo(n):
    # draw a number uniformly at random from 0, 1, 2, ..., n-1 in Theta(1)
    x = random.randrange(n)

    if x < math.log(n):
        for j in range(n**2):
            print(j)
    elif x < math.sqrt(n):
        print('Ok!')
    else:
        for i in range(n):
            print(i)
```

Problem 11.

State (but do not solve) the recurrence describing the runtime of the following function.

```
def foo(n):
    if n < 1:
        return 0

    for i in range(n**2):
        print("here")

    foo(n/2)
```

$$T(n) = \begin{cases} \Theta(1), & n = 1 \\ \boxed{}, & n > 1 \end{cases}$$

Problem 12. (2 points)

Solve the below recurrence, stating the solution in asymptotic notation. Show your work.

$$T(n) = \begin{cases} T(n/2) + \Theta(n) & n > 1 \\ \Theta(1) & n = 1 \end{cases}$$

Problem 13.

Suppose `bar` and `baz` are two functions. Suppose `bar`'s time complexity is $\Theta(n^3)$, while `baz`'s time complexity is $\Theta(n^2)$.

Suppose `foo` is defined as below:

```
def foo(n):
    if n < 1_000:
        bar(n)
    else:
        baz(n)
```

What is the asymptotic time complexity of `foo`?

Problem 14.

Let

$$f(n) = 5n \log n + \frac{n^3 + 5}{n + 2 + |\sin \pi n|} + n\sqrt{n}$$

Write f in asymptotic notation in as simplest terms possible.

$f(n) = \Theta($ $\square$ $)$

Problem 15.

Suppose $f_1(n)$ is $O(n^2)$ and $\Omega(n)$. Also suppose that $f_2(n) = \Theta(n^2)$.

Consider the function $f(n) = f_1(n) + f_2(n)$. True or false: it must be the case that $f(n) = \Theta(n^2)$.

- ◯ True
- ◯ False

Problem 16.

Suppose $f_1(n)$ is $O(n^2)$ and $\Omega(n)$. Also suppose that $f_2(n) = \Theta(n^2)$.

Consider the function $g(n) = f_2(n)/f_1(n)$. True or false: it must be the case that $g(n) = \Omega(n)$.

- ◯ True
- ◯ False

Problem 17.

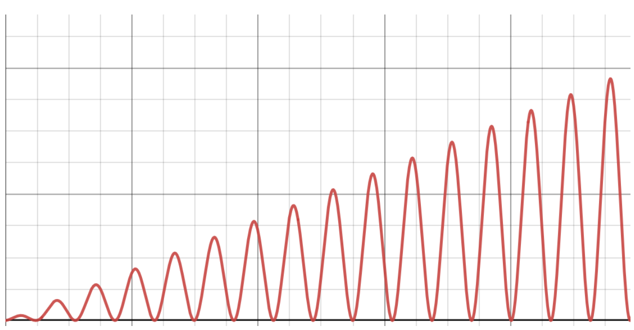
Suppose $f_1(n) = \Omega(g_1(n))$ and $f_2 = \Omega(g_2(n))$. Define $f(n) = \min\{f_1(n), f_2(n)\}$ and $g(n) = \min\{g_1(n), g_2(n)\}$.

True or false: it is necessarily the case that $f(n) = \Omega(g(n))$.

- ◯ True
- ◯ False

Problem 18.

Consider the function $f(n) = n \times (\sin(n) + 1)$. A plot of this function is shown below:



True or False: this function is $\Theta(n)$.

- ◯ True
- ◯ False

Problem 19.

Consider again the function $f(n) = n \times (\sin(n) + 1)$ from the previous problem.

True or False: $f(n) = O(n^3)$.

◯ True

◯ False

Problem 20.

Consider the iterative implementation of binary search shown below:

```
import math

def iterative_binary_search(arr, target):

    start = 0
    stop = len(arr)

    while (stop - start) > 0:
        print(arr[start])
        middle = math.floor((start + stop) / 2)
        if arr[middle] == target:
            return middle
        elif arr[middle] > target:
            stop = middle
        else:
            start = middle + 1
```

Which of the following loop invariants is true, assuming that `arr` is sorted and non-empty, and `target` is **not** in the array? Select all that apply.

☐ After each iteration, `stop - start >= 0`.

☐ After each iteration, `stop - start >= 1`.

☐ After each iteration, `arr[start] <= target`.

☐ After each iteration, `arr[start] <= max(target, arr[0])`.

Problem 21.

Consider `iterative_binary_search` from above and note the `print` statement in the `while`-loop. Suppose `iterative_binary_search` is run on the array:

```
[-202, -201, -200, -50, -20, -10, -4, -3, 0, 1, 3, 5, 6, 7, 9, 10, 12, 15, 22]
```

with target `11`.

What will be the last value of `arr[start]` printed?

☐

Problem 22.

Consider the code below which claims to compute the most common element in the array, returning a pair: the element along with the number of times it appears.

```python
import math

def most_common(arr, start, stop):
    """Attempts to compute the most common element in arr[start:stop]."""
    if stop - start == 1:
        return (arr[start], 1)

    middle = math.floor((start + stop) / 2)

    left_value, left_count = most_common(arr, start, middle)
    right_value, right_count = most_common(arr, middle, stop)

    if left_count > right_count:
        return (left_value, left_count)
    else:
        return (right_value, right_count)
```

You may assume that the function is always called on a non-empty array, and with `start = 0` and `stop = len(arr)`. Will this code always return the correct answer (the most common element)?

- ◯ Yes: it will always return the correct answer.
- ◯ No: it may recurse infinitely.
- ◯ No: it may try to access the array at an invalid index.
- ◯ No: it will run without error, but the element returned may not be the most common element in the array.

Problem 23.

Consider the modification of `mergesort` shown below, where one of the recursive calls has been replaced by an in-place version of `selection_sort`. Recall that `selection_sort` takes $\Theta(n^2)$ time.

```python
def kinda_mergesort(arr):
    """Sort array in-place."""
    if len(arr) > 1:
        middle = math.floor(len(arr) / 2)
        left = arr[:middle]
        right = arr[middle:]
        mergesort(left)
        selection_sort(right)
        merge(left, right, arr)
```

What is the time complexity of `kinda_mergesort`?

Problem 24.

Recall the partition operation from `quickselect`. Which of the following arrays could have been partitioned at least once? Select all that apply.

- ☐ `[50, 10, 20, 30, 60, 40]`
- ☐ `[20, 10, 30, 60, 40, 50]`
- ☐ `[20, 50, 10, 40, 30, 60]`
- ☐ `[60, 50, 40, 30, 20, 10]`
- ☐ `[10, 20, 30, 40, 50, 60]`

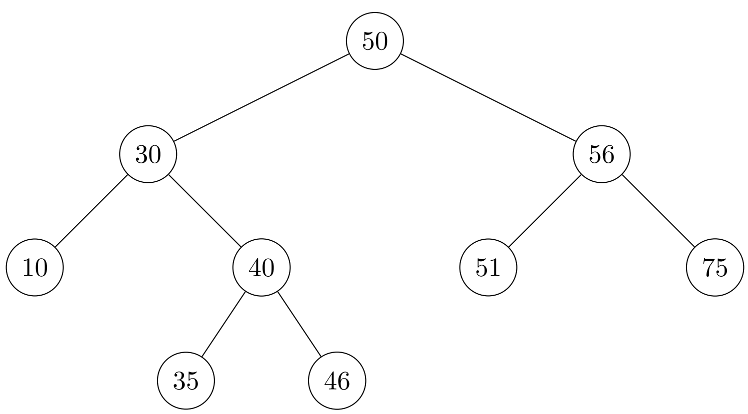
Problem 25.

Define the **largest gap** in a collection of numbers to be the largest difference between two distinct elements in the collection (in absolute value). For example, the largest gap in $\{4, 9, 1, 6\}$ is 8 (between 1 and 9).

Suppose a collection of n numbers is stored in a **balanced** binary search tree. What is the time complexity required for an efficient algorithm to calculate the largest gap of the numbers in the BST? State your answer as a function of n in asymptotic notation.

Problem 26.

Suppose the numbers `41`, `32`, and `33` are inserted (in that order) into the below binary search tree. Draw where the new nodes will appear.



Before turning in your exam, please check that your name is on every page.

(You may detach and use this page for scratch work. You do not need to turn it in.)

(You may detach and use this page for scratch work. You do not need to turn it in.)