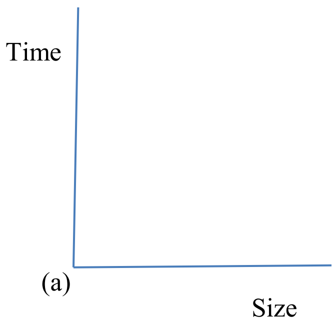
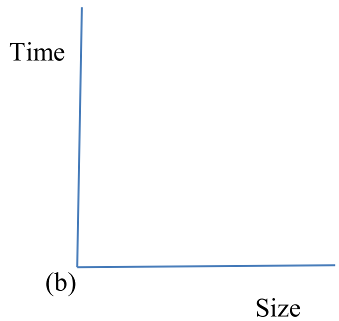
**IMPORTANT: Go to Gradescope and download the Quiz7 pdf file: print it, fill it in, and submit it! Do not fill in these pages!**

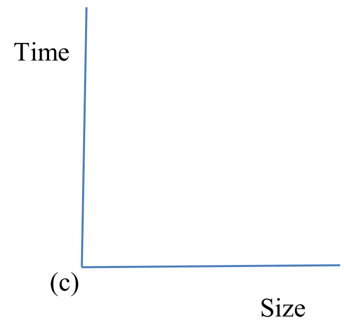
When working on this quiz, recall the rules stated on the Academic Integrity statement that you signed. There is no helper project file for this assignment. Submit your completed written quiz by 11:30pm on Friday (there is an In-Lab Exam on Thursday, so the due date is Friday). I will post my solutions to EEE reachable via the **Solutions** link on Saturday.

---

1. (3 pts) Sketch approximate Size vs. Time curves for the two algorithmic complexity classes required in each of the pictures below: for one, write **Impossible** instead: (a) an O(N) algorithm that is **always** faster than an O($N^2$) algorithm. (b) an O(N) algorithm that is **never** faster than an O($N^2$) algorithm. (c) an O(N) algorithm that is **sometimes** faster than an O($N^2$) algorithm.




2. (2 pts) Assume that a function **s** is in the complexity class $\mathbf{O(N\sqrt{N})}$. (a) What is its doubling-signature: how much more time (by what factor) does it take to solve a problem twice as large? Show your calculation and simplification to a numerical answer. (b) Briefly explain why it makes little sense for an algorithm to be in the complexity class **O(1/n)**?

3. (6 pts) Assume that functions **f1** and **f2** compute the same result by processing the same argument. Empirically we find that $\mathbf{T_{f1}(N) = 10\ N\ log_2\ N}$ and $\mathbf{T_{f2}(N) = 90N}$ where the times are in seconds. (a) Solve algebraically for what size **N** these two functions would take the same amount of time, showing how you **calculated** your answer. (b) for what size arguments is it better to use `f1`? `f2`? (c) Briefly describe how we can write a simple function **f** that runs as fast as the fastest of **f1** and **f2** for all size inputs. (d1) What exact integer value **N** (±1) solves $\mathbf{22\sqrt{N} = 10\ (Log_2\ N^2) + 10{,}000}$? Use a calculator, spreadsheet, or a program to guess and refine your answer (try plotting values to see where the curves meet). Your answer should be correct for all digits up to the ones-place: e.g., a number like 23,728. (d2) Based on your calculation, which complexity class $\boldsymbol{O(\sqrt{N})}$ or $\mathbf{O(Log_2\ N^2)}$ grows more slowly; justify why?

4. (6 pts) The following two functions each determine the distance between the two closest values in list `l`, with `len(l)` = N. (a) Write the complexity class of each statement in the box on its right. (b) Write the **full calculation** that computes the complexity class for the entire function. (c) Simplify what you wrote in (b).

```
def closest(l:[int])->int:
  a = set()
  for i in range(len(l)):
    for j in range(len(l)):
      if i != j:
        a.add(abs(l[i]-l[j]))
  return min(a)
```

```
def closest(l:[int])->int:
  a = sorted(l)
  min = None
  for i in range(len(a)-1):
    if min==None or a[i+1]-a[i]<min:
      min = a[i+1]-a[i]
  return min;
```

5. (5 pts) Assume that function **f** is in the complexity class **O($N^2$ ($\log_2$ N))**, and that for **N = 1,000** the program runs in **10 seconds**.

(1) Write a formula, **T(N)** that computes the approximate time that it takes to run **f** for any input of size **N**. Show your work/calculations by hand, approximating logarithms, then finish/simplify all the arithmetic.

(2) Compute how long it will take to run when **N = 1,000,000** (which is also written **$10^6$**). Show your work/calculations by hand, approximating logarithms, finish/simplify all the arithmetic. Compute the final result (time) not in **seconds** but in **days**.

6. (3 pts) Assume that we have recorded the following data when timing three methods (measured in milliseconds). Based on these times (which are measured and therefore approximate, so don't expect perfect results), **fill in an estimate** for the complexity class (one of the standard ones) for each method and **fill in an estimate** for the running time when N = **1,600**.

| N | Time: Method 1 | Time: Method 2 | Time: Method 3 |
|---|---|---|---|
| 100 | 300 | 20 | 20 |
| 200 | 604 | 76 | 22 |
| 400 | 1,196 | 325 | 20 |
| 800 | 2,395 | 1,178 | 19 |
| 1,600 | | | |
| Complexity Class Estimate | | | |