15-112 Spring 2023 Lecture 3/4
Quiz 6
20 minutes

Name: _____

Andrew ID: _____@andrew.cmu.edu

Section: _____

- **You may not use any books, notes, or electronic devices during this quiz.**
- **You may not ask questions about the quiz except for language clarifications.**
- **Show your work on the quiz (not scratch paper) to receive credit.**
- **If you use scratch paper, you must submit it with your andrew id on it, and we will ignore it.**
- **All code samples run without crashing unless we state otherwise. Assume any imports are already included as required.**
- **Do not use these topics: oop or recursion.**
- **You may use almostEqual() and rounded() without writing them. You must write everything else.**

1. Big-Oh [30 pts, 5 pts each]

For each of the following functions, state the worst-case big-oh runtime in terms of N, where N = len(L). Assume the functions never crash.

All the answers will be one of these:
O(1), O(logN), O(N), O(NlogN), $O(N^2)$, $O(N^2 \log N)$, $O(N^3)$, $O(2^N)$

Place your answers (and nothing else) in the box next to each block of code.

```
def f0(L):
    M = copy.copy(L)
    for i in range(len(L)):
        M[i] += i
        M.sort()
    return M
```

```
def f1(L):
    M = sorted(L)
    P = [ ]
    for i in range(len(L)):
        avg = sum(L) / len(L)
        if M[i] < avg:
            P.append(i)
    return P
```

```
# note: this is the same as f1(L), only here we compute avg just once:
def f2(L):
    M = sorted(L)
    P = [ ]
    avg = sum(L) / len(L)
    for i in range(len(L)):
        if M[i] < avg:
            P.append(i)
    return P
```

```
def f3(L):
    return L.pop(0) + L.pop()
```

```
def f4(L):
    i = len(L)-1
    t = 0
    while i > 0:
        t += L[i]
        i //= 2
    return t
```

```
def f5(L):
    L = copy.copy(L)
    M = [ ]
    N = len(L)
    for x in L:
        for y in L:
            P = [x*y] * N
            M.extend(P)
    while M != [ ]:
        L.append(M.pop())
    return L
```

2. More Big-Oh [30 pts, 5 pts each]
Answer the following. Be sure to show your work as appropriate.

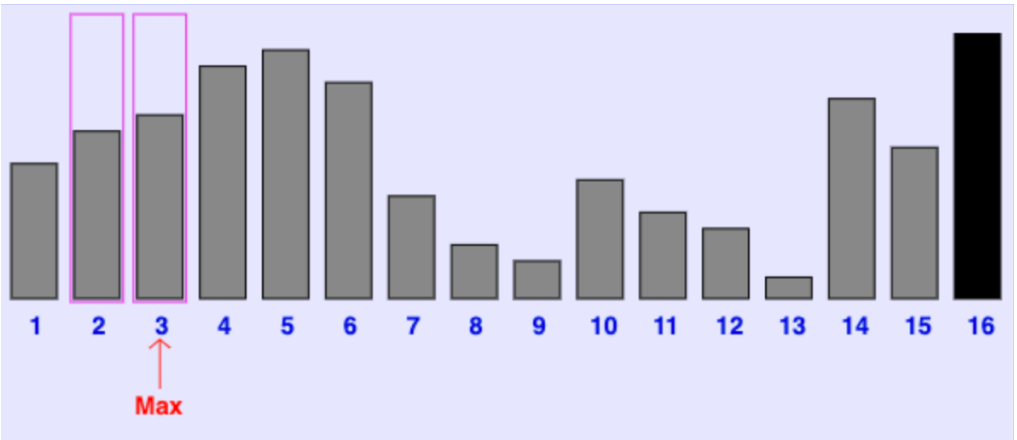
1. If a function f(L) runs in O(N**2) time, and f takes 2 seconds on a list of length 1000, how long will f take on a list of length 4000?

2. If a function g(L) takes 4 seconds on a list of length 300, and g takes 20 seconds on a list of length 1500, what is the most likely big-oh of g?

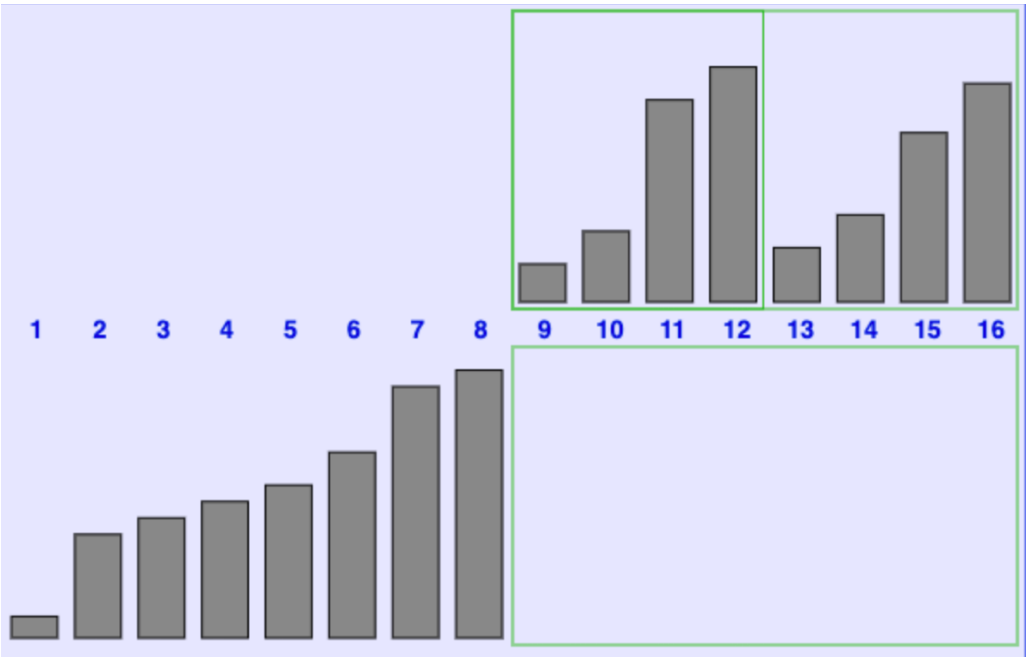
3. State the worst-case big-oh runtime of mergesort. Then, using a diagram like we did in lecture, or with words, explain why this is correct. You should be very brief, but you must make clear how many steps per pass there are, and how many passes there are.

4. Repeat the previous problem for selectionsort (state its worst-case big-oh runtime and explain why this is correct).

5. The following is a snapshot of xSortLab running selectionSort. What are the indexes of the next two values to be **swapped**?



6. The following is a snapshot of xSortLab running mergeSort. What are the indexes of the next two values to be **compared**?



3. FR: indexMap [40 pts]

Write the function indexMap(L) that takes a list L of strings and returns a dictionary d that maps each character in any string in L to a set of the indexes in L where a string containing that letter occurs.

For example:

```
L = [ 'aba', 'ac', 'bc' ]
assert(indexMap(L) == { 'a': {0, 1}, 'b': {0, 2}, 'c': {1, 2} })
```

4. Bonus [5 pts, 2.5 pts each]
Indicate what these print. Place your answers (and nothing else) in the box next to each block of code.

```
def f(L):
    M = copy.copy(L)
    while L:
        i = L.index(max(L))
        M *= L.pop(i)
    return sum(M)
print(f(list(range(1, 10, 3))))
```

```
# note: bin(n) is the string representation of n in base 2,
# preceded with '0b'.  So bin(6) is '0b110'
def g():
    def h(n):
        count = 0
        b = bin(n)
        for i in range(1, len(b)):
            count += int(b[i] == b[i-1] == '1')
        return count
    n = 0
    while h(n) != 3:
        n += 1
    while h(n) != 1:
        n += 1
    return n
print(g())
```