fullName: _____ andrewID: _____ section: ____

15-112 F25 Quiz5 Version B
Time: 25 Minutes

You must write your name on this paper and hand this back in immediately after the assessment. If we do not receive it immediately, you will receive a zero on the assessment. Do not unstaple any pages. All pages must be handed in intact.

Do not use your own scrap paper. You should not need it, but if you must absolutely have scrap paper, raise your hand and we will provide some. Write your andrewID clearly on it and hand it in with your quiz. We will not grade anything on scrap paper.

You may not ask questions during the quiz, except for English-language clarification questions. If you are unsure about a problem, take your best guess.

Before and during the quiz, you may not view any other notes, prior work, websites or resources, including any form of AI. You may not use calculators, phones, laptops, or any other devices. You may not communicate with anyone else except for current 112 TAs or faculty during the assessment. All syllabus policies apply.

You may not discuss this quiz with anyone else, even briefly, in any form, until we have released grades. Failure to abide by these rules may result in an academic integrity violation.

Do not use recursion or OOP.

Do not open this or look inside (even briefly) before you are ready to begin. Do not spend more than 25 minutes on this assessment.

Code Tracing [11 pts total, 5.5 pts each]

Indicate what the following code prints. Place your answer (and nothing else) in the boxes below. If a line of code crashes, just print "crash" (without quotes) and stop the CT at that point.

CT1:

```
def ct1(L):
    s = set(L)
    t = set([v for v in L if v%2 == 1])
    s.remove(len(t))
    t.add(len(s))
    return (s&t, s|t, s.difference(t))
print(ct1([1,2,3,4]))
```

CT2:

```
def ct2(n):
    d = { n : str(n) }
    for i in range(1, 5):
        k = i%3
        d[k] = str(i) + d.get(k, '')
    for k in d:
        if k == int(d[k]):
            d[k] *= k
    return d
print(ct2(0))
```

Crashes? (Yes/No) [15 pts total, 2.5 pts each]

For each of the following, bubble in Yes if it crashes and No if it does not crash. Do not bubble in both.

Assume S is a set with N integers.

Assume D is a dict with N keys where the keys are all integers.

1. `S[0]`

   ◯ Yes ◯ No

2. `D[S] = len(S)`

   ◯ Yes ◯ No

3. `S.add(S.add(type(S)))`

   ◯ Yes ◯ No

4. `S.add(S)`

   ◯ Yes ◯ No

5. `D[len(D)] = S`

   ◯ Yes ◯ No

6. `D.get('abc')`

   ◯ Yes ◯ No

True/False [20 pts total, 2.5 pts each]

For each of the following, bubble in True or False. Do not bubble in both.
Assume L and M are both lists with N integers.
Assume S and T are both sets with N integers.
Assume D is a dict with N keys where the keys are all integers.

1. `L` cannot be a key in `D`, but it can be a value in `D`.

◯ True ◯ False

2. `v in L` performs linear search even if `L` is sorted.

◯ True ◯ False

3. If `f(L)` performs an O(N**2) sort and `g(L)` performs an O(NlogN) sort, there cannot exist any list `L` where `f(L)` runs faster than `g(L)`.

◯ True ◯ False

4. If `sorted(L) != L`, then binary search on `L` will crash.

◯ True ◯ False

5. If `L == M`, then `set(L) == set(M)`.

◯ True ◯ False

6. If `len(set(L)) != len(set(M))`, then `L != M`.

◯ True ◯ False

7. If `len(S - T) == 0`, then `S == T`.

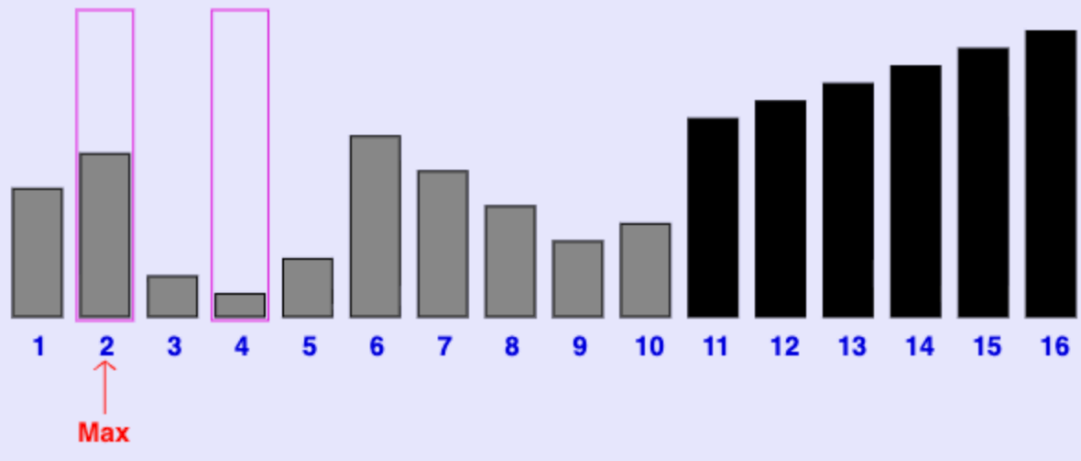
◯ True ◯ False

8. `L` can be an element in `S`.

◯ True ◯ False

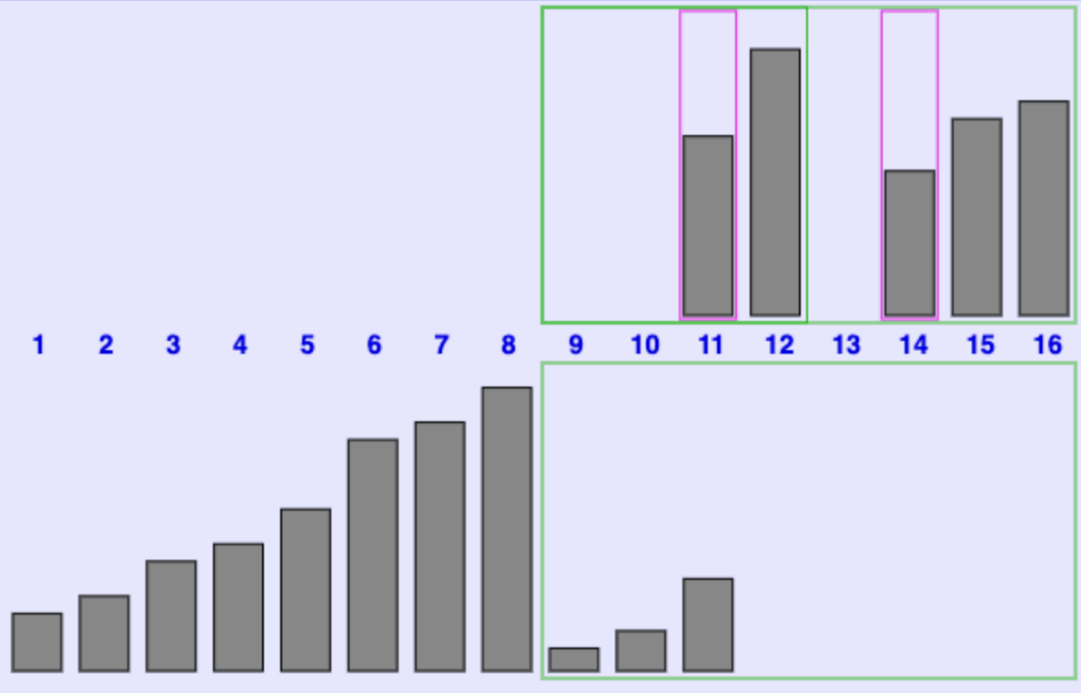
Searching, Sorting, and Hashing [36 pts total, 4 pts each]
Place your answer to each question in the box following that question.

1. The following image is from selection sort in xSortLab:



What are the indexes of the next two values to be swapped?

2. The following image is from merge sort in xSortLab:



What is the index of the next value to be copied to the temporary list?

3. When running selection sort over a list of N integers, how many steps are taken on the THIRD pass (where a step is either a compare or a swap)?

4. When running merge sort over a list of N integers, where you may assume that N is a power of 2...

A) How many steps per pass are required (where a step is either a compare or a copy)?

B) How many total passes are required?

5. If selection sort takes 4 seconds to sort 2 million values, how long (to the nearest second) would we expect selection sort to take to sort 6 million values on the same computer?

6. Heap sort is an O(NlogN) sort. Say we know these facts:

- Merge sort takes 8 seconds to sort a list L.
- Heap sort takes 5 seconds to sort the same list L.
- Merge sort takes 16 seconds to sort a list M.

Given these facts, how many seconds would we expect heap sort to take to sort the list M?

Note: for the next 2 questions:

- you may assume that 2**10 == 1,000, and
- we will accept answers within 2 of the correct answer.

7. For a list with 8,000 values, how many comparisons would be required in the worst case for linear search?

8. For a sorted list with 8,000 values, about how many comparisons would be required in the worst case for binary search?

9. For this question, assume:
1) hash tables are implemented as described in lecture,
2) each bucket in a hash table has a max bucket size of 2,
3) hash tables start with 4 buckets, and
4) `hash(x) == x` for any integer x.

Say we have this code:

```
s = set()
s.add(10)
s.add(44)
s.add(6)
s.add(10)
```

With the assumptions above, write the values in each bucket for the hash table that represents this set after running those lines of code:

0: _____

1: _____

2: _____

3: _____

Big O [18 pts total, 3 pts each]

For each of the following, assuming N is a positive integer, and assuming L is a list of length N, indicate the big-O of the function (that is, of its worst-case run time).

Bubble in the correct answer to the left of each function. Do not bubble in more than one response per function.

◯ O(1)
◯ O(logN)
◯ O(N**0.5)
◯ O(N)
◯ O(NlogN)
◯ O(N**2)
◯ O(2**N)

```
def f1(N):
    result = 0
    while N > 0:
        N -= 2
        result += 1
    return result
```

◯ O(1)
◯ O(logN)
◯ O(N**0.5)
◯ O(N)
◯ O(NlogN)
◯ O(N**2)
◯ O(2**N)

```
def f2(N):
    result = 0
    while N > 0:
        N //= 2
        result += 1
    return result
```

◯ O(1)
◯ O(logN)
◯ O(N**0.5)
◯ O(N)
◯ O(NlogN)
◯ O(N**2)
◯ O(2**N)

```
import math
def f3(N):
    result = 2
    # note: math.log(x, 2) returns the log of x in base 2
    while math.log(result, 2) < N:
        result += 1
    return result
```

- ◯ O(1)
- ◯ O(logN)
- ◯ O(N**0.5)
- ◯ O(N)
- ◯ O(NlogN)
- ◯ O(N**2)
- ◯ O(2**N)

```
def f4(L):
    result = 0
    M = L[10:20]
    for i in range(1, len(L), len(L)//10):
        result += sum(M)
    return result
```

- ◯ O(1)
- ◯ O(logN)
- ◯ O(N**0.5)
- ◯ O(N)
- ◯ O(NlogN)
- ◯ O(N**2)
- ◯ O(2**N)

```
def f5(L):
    s = set(L)
    d = dict()
    for v in L:
        d[v] = s
    return d
```

- ◯ O(1)
- ◯ O(logN)
- ◯ O(N**0.5)
- ◯ O(N)
- ◯ O(NlogN)
- ◯ O(N**2)
- ◯ O(2**N)

```
def f6(L):
    M = [ ]
    for v in L:
        M.insert(0, v)
    return sorted(M)
```

BonusCT [2pts]

This CT is optional, and intended to be very challenging. It is worth very few points. Indicate what the following code prints. Place your answer (and nothing else) in the box below. If a line of code crashes, just print "crash" (without quotes) and stop the CT at that point.

```
def bonusCt(s):
    s = ''.join(sorted(set(s.upper()))).strip()
    c = min(s)
    e = eval(str({3}).replace(str(ord('f')-ord('c')),str()))
    for i in range(len(s)):
        d = chr(ord(s[::-1][i]) - ord(c) + ord('c'))
        d = d if d <= 'e' else 'e'
        e[d] = i
    return e

print(bonusCt('High hopes'))
```