Question - 1
Weighted Quick Union 1

SCORE: **5 points**

In the "Weighted Quick Union" algorithm, we link the root of the smaller tree to the root of larger tree at every union operation. In the worst case, for N elements, what's the maximum depth of all leaves? **[The depth of a single node is 0]**

- N
- sqrt(N)
- lgN+1
- [✓ lgN]

Question - 2
Weighted Quick Union_1

SCORE: **5 points**

How many array accesses at most for the weighted quick-union algorithm to process M connections among N sites? **[Ignore the coefficient and the depth of a single node is 0]**

- [✓ M * lgN]
- M * N
- M * lgN + N
- M * (lgN + 1)

Question - 3
Quick Find Vs. Quick Union

SCORE: **5 points**

Choose all the correct statements regarding the quick-find and quick-union algorithms:

- [✓ The quick-find algorithm at least uses N + 3 array accesses for each call to union() operation on N objects.]
- [✓ The tree formed in quick-find is flat while the tree formed in quick-union can get tall.]
- We can guarantee it that using quick-union for dynamic connectivity problem is substantially faster than using quick-find in every case.

Help

○ If we use quick-find for the dynamic connectivity problem and wind up with a single component. This requires at least N -1 calls to union(), and, consequently, at most N^2 array accesses.

Question - 4

Union-Find Implementations Analysis

SCORE: 5 points

Please choose all correct statements regarding description of different union-find implementations:

○ Weighted quick union with path compression is optimal and quite constant-time per operation.

[✓ For weighted quick union algorithm: union() operation will just take constant time, given roots.]

[✓ The number of array accesses used by find() in quick-union is 1 plus the twice the depth of the node corresponding to the given site.]

[✓ Suppose that the input pairs come in the order 0-1, then 0-2, then 0-3, and so forth. After N - 1 such pairs, the tree that is formed by the quick-union algorithm may have height N - 1.]

Question - 5

Linked List

SCORE: 5 points

Suppose x is a linked-list node and not the last node on the list. What is the effect of the following code fragment?

```
x.next = x.next.next;
```

○ Deletes the node x from the list

[✓ Deletes the node immediately following x from the list]

○ Deletes the node immediately following x.next;

○ None of the above

Question - 6

Stack and Queue

SCORE: 5 points

Suppose that you have empty two data structures: a stack and a queue. Items can be pushed on to the stack and, at any time, the item on the top of the stack may be popped and enqueued into the queue.

The following items: *a, b, c, d, e,* and *f* are pushed onto the stack in the order given. Which of the following sequences can *never* be the state of the queue after the stack is empty? Note that the left-most item shown is the oldest element of the queue.

- f e d c b a
- b c a f e d
- d c e f b a
- [✓ c a b d e f]

Question - 7
Array vs. Linked List

SCORE: **5 points**

Which of the following points is/are true about a Linked List data structure when compared with an array?

- Array has better cache locality that can make them better in terms of performance.
- Random access is not allowed in the typical implementation of a linked list.
- The size of array has to be pre-decided, linked lists can change their size any time.
- [✓ All of the above.]

Question - 8
Weighted Quick Union 2

SCORE: **5 points**

Which is the correct content of id[] array when you use weighted quick-union for the following sequence of union operations on a set of 10 items?

```
    9-0     3-4     5-8     7-2     2-1     5-7     0-3
4-2
```

Assume the original array is of the size 10, like below:

```
   [ 0   1   2   3   4   5   6   7   8   9 ]
```

If two components have the same size (weight), then link the first component into the second component.

- 1 1 4 4 1 8 6 2 1 0
- 1 1 1 4 0 8 6 2 1 0
- [✓ 4 2 2 4 2 8 6 2 2 0]
- 4 2 2 4 2 8 6 2 1 0