Q1. Runtime analysis Jason Qi

a). let k be the number of iteration performed:

k	1	2	3	4	...	m
i	2	4	16	256		
	$= 2^1$	2^2	$(2^2)^2$	$((2^2)^2)^2$		2^{2^m}

$\therefore T(n) = \sum_{k=0}^{m} O(1)$, where m is the largest integer such that $2^{2^m} < n$.

$$2^{2^m} \approx n \Rightarrow m = \log_2(\log_2 n)$$

$$T(n) = \sum_{k=0}^{\log_2(\log_2 n)} O(1)$$

$$= O(\log(\log n))$$

$$\therefore T(n) = \Theta(\log(\log n))$$

b). inner loop: $k = 0$ to $k < i^3$

$$\therefore \sum_{k=0}^{i^3-1} \theta(1)$$

outer loop: $i = 1$ to $i \leq n$

$$\therefore \sum_{i=1}^{n} (\text{inner chunk})$$

$$\therefore T(n) = \sum_{i=1}^{n} \left(\theta(1) + O\left(\sum_{k=0}^{i^3} \theta(1)\right)\right)$$

If statement: $i \% \sqrt{n} = 0$

let x be the number of iterations when the if statement is true:

x	1	2	3	4	...	m
i	$\sqrt{n}$	$2 \cdot \sqrt{n}$	$3 \cdot \sqrt{n}$	$4\sqrt{n}$		n

$\therefore$ stops when $i = n = m \cdot \sqrt{n}$

$$n = m \cdot \sqrt{n}$$

$$m = \sqrt{n}$$

$$T(n) = \sum_{i=1}^{n} \theta(1) + \sum_{x=1}^{\sqrt{n}} \sum_{k=0}^{i^3-1} \theta(1)$$

Since code only runs when $i = x \cdot \sqrt{n}$

$$= \theta(n) + \sum_{x=1}^{\sqrt{n}} \theta((x \cdot \sqrt{n})^3)$$

$-$

$$= \Theta(n) + \sqrt{n}^3 \sum_{x=1}^{\sqrt{n}} \Theta(x^3)$$

Since $\sum_{i=1}^{n} \Theta(i^p) = \Theta(n^{p+1})$

$$= \Theta(n) + \sqrt{n}^3 \cdot \Theta((\sqrt{n})^4)$$

$$= \Theta(n) + \Theta(n^{\frac{3}{2}} \cdot n^{\frac{4}{2}})$$

$$= \Theta(n) + \Theta(n^{\frac{7}{2}})$$

$$\therefore T(n) = \Theta(n^{\frac{7}{2}})$$

c). inner loop: from $m=1$ to $m \leq n$, $m+m$ each iteration.

k	1	2	3	4	5	x
m	1	2	4	8	16	2^{x-1}
	2^0	2^1	2^2	2^3	2^4	

$\therefore \sum_{m=1}^{x} \Theta(1)$, where $2^{x-1} \leq n$

$$2^{x-1} \approx n$$

$$x \approx \log_2 n - 1$$

$$T(n) = \sum_{m=1}^{\log n - 1} \Theta(1) = \Theta(\log n)$$

middle loop: from $k=1$ to $k \leq n$, k ++

$$\sum_{k=1}^{n} (\text{inner chunk})$$

outer loop: from $i=1$ to $i \leq n$, i ++

$$\sum_{i=1}^{n} (\text{code chunk})$$

$$\therefore T(n) = \sum_{i=1}^{n} \sum_{k=1}^{n} (\Theta(1) + O(\Theta(\log_2 n)))$$

If statement true when the k^{th} element in the array is equal to i.

- Worst case → every element in the array match with one (i, k) pair.

$$T(n) = \Theta(n^2 \cdot \log n)$$

- best case → no matches

$$T(n) = \Theta(n^2)$$

d). inner loop: from $j=0$ to $j < size$, $j{+}{+}$

$$\therefore T(n) = \sum_{j=0}^{size-1} \theta(1)$$

outer loop: $T(n) = \sum_{i=0}^{n-1} (\text{ inner chunk})$

If statement true when $i == size$,
let k be number of iterations if() is true:

k	1	2	3	4	...	m
size	10	$\frac{3\cdot 10}{2}$	$\frac{3\cdot 10}{2}\cdot\frac{3}{2}$	$\left(\frac{3}{2}\right)^3\cdot 10$		$\left(\frac{3}{2}\right)^{m-1}\cdot 10$

Considering the number of operation we need for new[] & delete[],
we get:

$$T(n) = \sum_{i=0}^{n-1} \left(\theta(1) + \theta\left(\sum_{j=0}^{size-1} \theta(1) \right) + \sum_{x=1}^{m} \theta(1) \right.$$

$$= \theta(n) + \sum_{k=1}^{m} \left(\sum_{j=0}^{size-1} \theta(1) \right) + \theta(m)$$

Since $n \geq m = \left(\frac{3}{2}\right)^{m-1} \cdot 10$

$n \approx \left(\frac{3}{2}\right)^{m-1} \cdot 10$

$\frac{n}{10} \approx \left(\frac{3}{2}\right)^{m-1}$

$m = \log_{\frac{3}{2}} \cdot \left(\frac{n}{10}\right) + 1$

$$= \theta(n) + \sum_{k=1}^{m} \left(\theta(size) \right) + \theta(\log n)$$

Since for each iteration k, $size_k = 10 \cdot \left(\frac{3}{2}\right)^k$

$$= \theta(n) + \sum_{k=1}^{m} \left(\theta\left(10 \cdot \left(\frac{3}{2}\right)^k \right) \right) + \theta(\log n)$$

Since $\sum_{i=0}^{n} c^i = \frac{c^{n+1} - 1}{c - 1}$

$$= \theta(n) + 10 \cdot \frac{\left(\frac{3}{2}\right)^{\log_{\frac{3}{2}} \cdot \left(\frac{n}{10}\right) + 1} - 1}{\left(\frac{3}{2}\right) - 1} + \theta(\log n)$$

$$= \theta(n) + 10 \cdot \frac{\frac{n}{10} \cdot \frac{3}{2} - 1}{1} + \theta(\log n)$$

$$T(n) = \theta(n) + \theta(n) + \theta(\log n)$$

$$\therefore T(n) = \theta(n)$$

Q2. Recursion tracing

a).

```
struct Node {
    int val;
    Node*  next;
};

Node* llrec(Node* in1, Node* in2)
{
    if(in1 == nullptr) {
        return in2;
    }
    else if(in2 == nullptr) {
        return in1;
    }
    else {
        in1->next = llrec(in2, in1->next);
        return in1;
    }
}
```

set in1 as the header & "attach" rest of the linked list to in1.

1, 5, 2, 6, 3, 4

llrec (in1 , in2)
llrec (1 , 5)

5, 2, 6, 3, 4

llrec (in2 , in1 -> next)
llrec (5, 2)

2, 6, 3, 4

llrec (in1 -> next , in2 -> next)
llrec (2, 6)

6, 3, 4

llrec (in2 -> next , in1 -> next -> next)
llrec (6, 3)

3, 4

llrec (in1 -> next -> next , in2 -> next -> next)
llrec (3, nullptr)

∴ Final output : 1, 5, 2, 6, 3, 4

b). llrec (nullptr , 2)

$\to$ 2

$\therefore$ output: 2