if $f(n)$ is $O(g(n))$,

$c \in \mathbb{R}, n_0 \in \mathbb{Z} \mid n_0 \geq 1 \mid f(n) \leq c g(n)$ for $n \geq n_0$

Q1 a) $f(n) = \log_{10}(n^2)$

$= 2\log_{10}(n)$

Show $f(n)$ is $O(\log_2(n))$

$\log_2(n) = \frac{\log n}{\log 2} = \frac{1}{\log 2} \log_{10}(n)$

$\frac{1}{\log_{10}(2)} > 2$, so for any $n \geq 1$,

$\log_{10}(n^2) \leq \log_2(n)$

$(n_0 = 1, c = 1)$

$y = \log_{10} x$

$x = 10^y$

~~$c \in \mathbb{R}, n_0 \in \mathbb{Z}$ $f(n)$ is $\Omega(g(n))$ if:~~

$c \in \mathbb{R}, n_0 \in \mathbb{Z} \mid c > 0, n_0 \geq 1, \quad f(n) \geq c \cdot g(n)$ for $n \geq n_0$

b) $f(n) = n(10n^2 - 2\sqrt{n}) = 10n^3 - 2\sqrt{n^3} = 10n^3 - 2n^{3/2}$

$g(n) = n^3$

$10n^3 \geq 10n^3 \quad n \geq 0$

$-2n^{3/2} \geq -2n^3 \quad n \geq 1$

$10n^3 - 2n^{3/2} \geq 10n^3 - 2n^3$ for $n \geq 1$

$f(n) \geq 8 \cdot n^3$ for $n \geq 1$

$f(n) \geq 8 \cdot g(n)$ for $n \geq 1$

So $c = 8, n_0 = 1$. $f(n)$ is $\Omega(n^3)$

$c_1, c_2 \in \mathbb{R}, n_0 \in \mathbb{Z} \mid c_1 g(n) \leq f(n) \leq c_2 g(n)$ for $n \geq n_0$

c) $f(n) = (n \sin(n))^2 + 100$

$g(n) = n^2$

$c_1 g(n) \leq f(n)$

$n^2 \leq (n \sin(n))^2 + 100$

$n^2 \nleq n^2 \sin^2(n) + 100$

due to oscillating nature of $\sin(n)$, $f(n)$ periodically has value of 100 ($\sin(n) = 0$). As such, $c_1 g(n) \leq f(n)$ expression cannot find c_1 value to satisfy $c_1 g(n) \leq f(n)$. So $f(n)$ is NOT $\Theta(n^2)$

$f(n) \leq c_2 g(n)$

$n^2 \sin^2(n) \leq n^2$ for $n \geq 0$

$100 \leq 100 n^2$ for $n \geq 1$

$(n \sin(n))^2 + 100 \leq 101 n^2$ for $n \geq 1$

$c_1 = 101, n_0 = 1$

Q2 a)

worst case: loop runs until condition $i^2 \leq n$ is broken.

If such case happens, $i^2 > n$ is true, and so $i > \sqrt{n}$.

So, worst run $\sqrt{n}$ (-2) (+1) $\rightarrow \sqrt{n} - 2 + 1 = \sqrt{n} - 1$, $O(\sqrt{n})$

$i = 2$ initially; for when $i == \sqrt{n}$; worst running time for input n

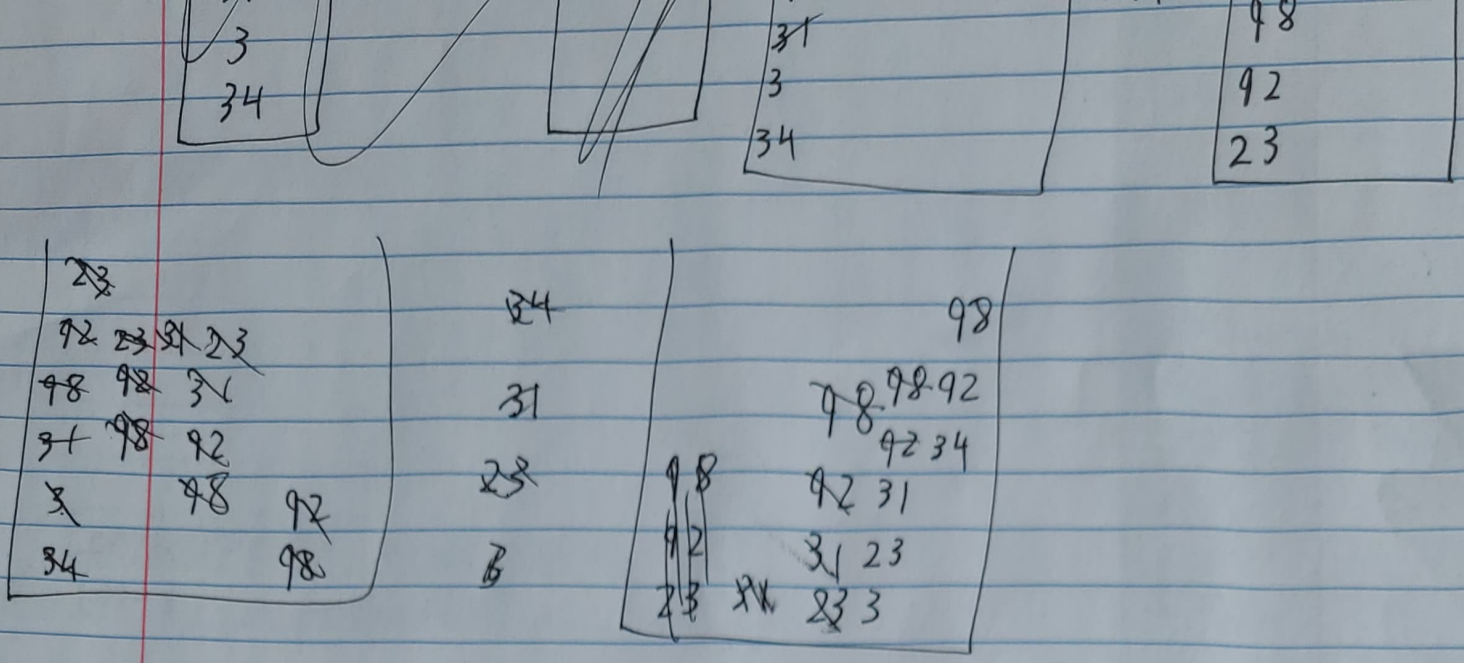
b) Best case: Immediately exiting loop via if ($n \% i == 0$). Since i initially equals 2, n must thus is multiple of 2, $n = 2k; k \in \mathbb{Z}$ (even number). This would comprise of only 1 iteration $O(1)$

$n = 2k; k \in \mathbb{Z}$

$n \% i = 2k \% i$

Initially, $i = 2$

$2k \% 2 = 0$



Q3 a)

```
while (! in.isEmpty()){
    temp = in.pop();
    while (!tmp.isEmpty() ∧ tmp.peek() > temp){
        in.push(tmp.pop());
    }
    tmp.push(temp);
} //d/h
while (!tmp.isEmpty()){
    in.push(tmp.pop());
}
```

h)

$O(n^2)$

worst case: input is (bottom to top) increasing. Each popped elem from input would require pushes back to input stack; $n \times n = n^2$

So,

$O(n^2)$

Q4 a) //note: most for loops are because I didn't want to write same line 3 times

```
~~for (int i=1; i≤8; i++){~~
for (int i=1; i≤3; i++){
    Q.enque(D.removeFirst());
}
for (int i=1; i≤3; i++){
    Q.enque(D.removeLast());
}
    Q.enque(D.removeFirst());
    Q.enque(D.removeLast());
for (int i=1; i≤8; i++){
    D.addLast(Q.deque());
}
for (int i=1; i≤8; i++){
    Q.enque(D.removeFirst());
}
~~D.enqu~~
D.addFirst(Q.deque());
for (int i=2; i≤4; i++){
    D.addLast(Q.deque());
}
for (int i=1; i≤3; i++){
    D.addFirst(Q.deque());
}
return(D);
```

b) O(n). no nested loops are present, and the largest number of ~~loops~~ executions by one loop is 8 (size of input). so, algorithm is O(n)

c) O(n). supposing D is given, we are using initially empty queue, Q, to perform algorithm. Q is the only variable used in algorithm, and may only contain the same number of elements as D (n). so, O(n).