problem 2.

part (a): Runtime = $\Theta(n^2)$

```
for (int i=0 ; i <n; i++)
    for (int j=0=i ; j< n ; j++) {
        S.push(j);
    }
}
```

1) $\sum_{i=0}^{n} i = n + n-1 + \cdots 1 = \frac{(n+1)\,n}{2} = \Theta(n^2)$

It's the same when it calls S.pop()

2) $\sum_{i=0}^{n} i = n + n-1 + \cdots 1 = \frac{(n+1)\,n}{2} = \Theta(n^2)$

3) for while loop
Because of the previous pop() and push this never even get called.

∴ Total = $\Theta(n^2) + \Theta(n^2)$

= $\Theta(n^2)$

Problem 2.

part (b)

when func(0, n) is called

↓

fun(n-1, n-1)

when curr = n-1,

it will call func(curr-1, n) to reduce curr to 0

and then call func(n-1, n-1) to start over, but

the time of invitation reduces 1.

$$\sum_{i=1}^{n}(n-i) = n-1 + n-2 + \cdots 1$$

$$= \frac{n(n-1)}{2} = \Theta(n^2)$$

problem 2

part (c)

i) for (int i=1 ; i <= n ; i++) {
    q.enqueue(i)
}

$$\sum_{i=0}^{n} i = n$$

ii) For the while loop,

Because if when (snap == true) $\wedge$ (q.front == 1)
it will run the for-loop and dequeue and delete "1" from the queue.
Therefores, the for loop will only run once.

∴ In total, in order to eliminate the queue, the total number of elements should be delete will be $2n$.

∵ To delete one element from queue,
it will have to run through both if and else, so three $\theta(1)$ statement to delete one elem-

∴ Total = $n + 1 + a + 2n \cdot 3$
= $8n + 1$
= $\theta(n)$

problem 2
part (d)

1) Creating the linked ~~list~~ list

∵ three statements per iteration
∴ $t_1 = 3n$

2) for the for loop:
it will go through n times with $i = 1 \to n$
and every time it will go through the list from the head to the tail.

and ∵ if ((curr → value % i == 0) && (arr[i] == 0))

∴ The time that the inner for loop will run will be $n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \cdots 1$

∵ This is the worst case for arr[j]
∴ assume arr[1] is always 0.
∴ the run time for the inner loop will always be n^2 every time

∴ Total $= 3n + n^2 \left(n + \frac{n}{2} + \frac{n}{3} + \cdots 1 \right)$

$= 3n + n^2 (\ln n)$

$= \theta (n^2 \ln n)$

$= \theta (n^2 \log n)$

Problem 3

part (a)

The worst case runtime for some func is when bar() is called in the function.

$\therefore$ Runtime = $\Theta(n^2) + \Theta(1) + \Theta(1)$

$= \Theta(n^2)$

$\therefore$ The worst case runtime is $\underline{\underline{\Theta(n^2)}}$

part (b)

when $n = 0$ and $max = 1$

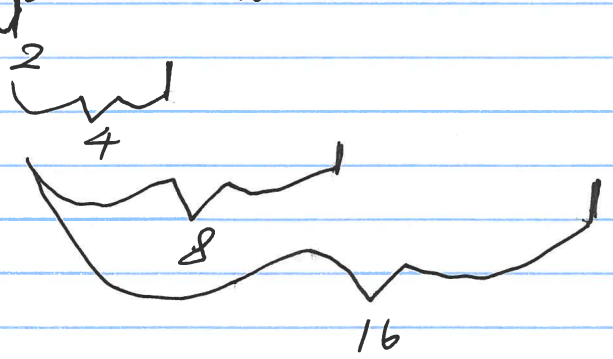
the function will increment n every time it goes into the some func()

when $n \neq max$, it will call foo() and increment n

when $n = max$, it will call bar() and double max

Therefore, when it's called n^{th} time

the call tree is



$\therefore$ When nth call, foo() will be called $n - \log n$ times

bar() will be called $\log n$ times

Amotised runtime = $\dfrac{(n - \log n)\log n + \log n^2}{n}$

$= \log n - \dfrac{\log^2 n}{n} + n \log n$

$= \Theta(n \log n)$

part (c)

It's similiar to the part (c), but now bar() has $\Theta(n^2)$ too takes $\Theta(n \log n)$

$$\text{armotised runtime} = \frac{(n - \log n)\, n \log n + n^2 \log n}{n}$$

$$= n \log n + \log n^2 + n \log n$$

$$= n \log n$$

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