Problem 1

Angel F.

enqueue 1 : S1 : 1
S2 :

enqueue 2 : S2 : 1, 2
S2 : 2

dequeue : S1
S2 : 2

enqueue 3 S1 : 3
S2 2

enqueue 4 S1 : 3, 4
S2 : 2

dequeue S1 : 3, 4
S2 :

enqueue 5 : 3, 4, 5

enqueue 6 : S1 : 3, 4, 5, 6 (bottom, top)
S2 :

Part b:

The worst-case scenario for enqueue is O(1) b/c push takes O(1) and enqueue only pushes one item.

The worst case scenario for dequeue is when stack 2 is empty and you have to pop and push items from stack 1 to stack two.

If you have n items in stack one it'll take 2·O(1) to push to stack 2 and pop from stack 1. ∴ if you are doing 2·O(1) n times, it'll take $2n = O(n)$.

Part C

For enqueue, pushing and popping will always be done at constant time.
For dequeue you will only get n when it is empty. So say if you have a really large size n in stack 1 until the point where stack 2 is empty, popping stack 2 will be O(1) n times. When you get the empty stack, it will take O(n) to push and pop items from stack 1 to stack 2 but then you'll also have O(1) pop on stack 2. Eventually, O(1) will override the price of O(n) making the amortized runtime of enqueue and dequeue O(1).

Problem 1

Angel F.

Part 1

The worst case scenario for enqueue would still be O(1), because push takes O(1) time. For dequeue the worst case scenario occurs when there is an empty stack 2 and you need to pop and push everything from stack 1. From Part (b) we know that if we push O(1) n times we get O(n), now if the price of popping stack 1 is O(n), we have to run O(n) n times making the worst case runtime be $O(n^2)$.

Now for amortized, we only pay the price of $O(n^2)$ when stack 2 is empty, however when it is not, we have n objects that we have to pop at O(1)

$$A_n^3 = \frac{\overbrace{O(n^2)}^{\text{empty list}} + \overbrace{O(n)}^{\text{non empty}}}{n_{\text{total}}} = A_n^3 = O(n)$$

Angel Flory

Problem 2

```
else if ((x%(int)Sqrt(n))==0){
    for (int i=1; i<=n; i++){
        for (int j=0; j<i; j++){
            O(1)
        }
    }
}
```

$\sum_{j=0}^{i} (1) = i$

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

n=100 $\sqrt{n}=10$

x=00

100%10=0
90%10=0
80%10=0
70%10=0
60%10=0
10 times

x=99

n=144 $\sqrt{n}=12$

144%12
132%12
12 times

$\therefore$ x%(int)Sqrt(n) runs $\sum_{0}^{\sqrt{n}} k = \sqrt{n}$

From here, I saw that the else if will run at most $\sqrt{n}$ times

$\therefore$

$\sum_{k=0}^{\sqrt{n}} \sum_{i=0}^{n} \sum_{j=0}^{i} O(1) = \sum_{k=1}^{\sqrt{n}} k^2 = \frac{(\sqrt{n})(\sqrt{n}+1)(2\sqrt{n}+1)}{6}$

$= n \cdot \sqrt{n} = \Theta(n^{3/2})$