Amortized Analysis Homework

I counstructed a C++ program to research this problem, which can be found in this folder

Problem 1

Right side represents the stack top

1. Step 1

stack A	5

stack B

2. Step 2

stack A	7	5

stack B

3. Step 3

- pop from stack A, push to stack B

stack A

stack B	5	7

- pop from stack B

stack A

stack B	7

4. Step 4

stack A	**3**

stack B	**7**

5. Step 5

stack A	**3**	**6**

stack B	**7**

6. Step 6

stack A	**3**	**6**	**9**

stack B	**7**

7. Step 7

- pop from stack A, push to stack B. until stack A is empty

stack A

stack B	**9**	**6**	**3**	**7**

- pop from stack B

stack A

stack B	**6**	**3**	**7**

8. Step 8

stack A

stack B	**3**	**7**

9. Step 9

stack A

stack B	7

10. Step 10

stack A

stack B

Here is a sample output:

```
Stack A: 5
Stack B:

Stack A: 7 5
Stack B:

Stack A:
Stack B: 7

Stack A: 3
Stack B: 7

Stack A: 6 3
Stack B: 7

Stack A: 9 6 3
Stack B: 7

Stack A: 9 6 3
Stack B:

Stack A:
Stack B: 6 9

Stack A:
Stack B: 9

Stack A:
Stack B:
```

Problem 2

As cost of a push or a pop cost 1 unit of work.

- push into queue:
  - push x onto stack A: 1 unit

In total: 1 unit

> n represents the number of elements in the stack A
> the number of elements in stack B is irrelevant

- pop from queue:
    - if B is not empty:
        - pop from stack B: 1 unit
    - if B is empty:
        - pop from stack A (n units), push to stack B (n units), until stack A is empty: 2n units
        - pop from stack B: 1 unit

In total: 1 unit if B is not empty, 2n + 1 units if B is not empty

In case that all enqueue operations are grouped together,
The dequeue operation costs $2n + 1$ units with n cost on popping A, and n cost pushing into B, finished with B getting popped once, So the total cost is $2n + 1$ units.

Problem 3

explain how a simplistic worst-case analysis would lead to the conclusion that after n operations, O(n^2) units of work would have been done

In a simplistic worst case, in the worst case scenario, a step would cost $2n + 1$ or $2n - 1$ (needs to be transfered from A to B to be dequeued. So the total cost is $n(2n + 1)$ units. This result in a total of O(n^2).

Problem 4

Using the accounting method, assign the lowest (integer) amortized cost possible to the enqueue operation and to the dequeue operation so that after any sequence of n1 enqueues and n2
dequeues, n1 >= n2, the amortized cost is >= the actual cost. Explain why this works

One possibility is enqueue = 3 and dequeue = 1.

Given that enqueuing x on stack A cost 1, giving a credit of 2 with item x. If x has to be transfered from A to B, it will cost 2 units, leaving 1 credit for x. Popping x from B costs 1. So the total cost is equal to or less than the amortized cost.

Problem 5

If amortized cost of each operation is constant, it will be $nO(1) = O(n)$ in total for n operations.