CS 1181 - Computer Science II

Practice Problem: Tiled Task

Purpose: To become familiar with stacks and queues.

Part A:

Write a main program that creates both a stack of integers (using class ArrayDeque) and a queue of integers (also using class ArrayDeque). Add the following integers in this order: 9, 6, 4, 2, 5, 1, 3.

For each data structure, pop/poll the next value and print it until the data structure is empty.

Part A example output:

```
Stack contents:
3
1
5
2
4
6
9

Queue contents:
9
6
4
2
5
1
3
```

Part B:

You are playing a simple game with a stack of tiles numbered 1 through 3 and a queue containing a pool of additional tiles. Each turn you look at the tile on the top of the stack and perform the appropriate operation:

- If the tile is a 1, you remove that tile plus one more from the top of the stack.
- If the tile is a 2, you remove that tile plus two more from the top of the stack.
- If the tile is a 3, you remove that tile but take the three next tiles from the queue and add them to the top of the stack.

When the player's stack has become empty, the game is over. If this happens in the middle of a turn (for example, if the player is supposed to remove three tiles from the stack but there are only two tiles left), that still counts as a turn. You can assume that the queue will always contain enough tiles to reach the end of the game.

Write a class called TileGame that contains a method that takes the stack and queue of integers and returns the number of turns it takes for the player's stack of tiles to become empty. Your method must have the following signature:

```
public static int tileGame(ArrayDeque<Integer> stack, Queue<Integer> q)
```

Your program must be able to handle all test cases without causing an exception in order to receive full credit, so be sure to try to think of tricky cases and check them.

Note that this problem does not require recursion to solve (though you can use that if you wish).

Example of game execution: If the queue contains the items 1, 2, 2, 1, 3, 1, 2, 1, 2, 3

and the stack contains (from top to bottom) the items 3, 2, 1, 2

then the return value of your method should be **3** because the game will proceed as follows:

- Turn 1: discard the 3 and push 1, 2, 2 so the stack is now 2, 2, 1, 2, 1, 2
- Turn 2: discard the 2 as well as the following 2, 1 so the stack is now 2, 1, 2
- Turn 3: discard the 2 and the following 1, 2, so the stack is now empty

Part B example output:

```
Given stack and queue took 3 turns to finish playing

Given stack and queue took 5 turns to finish playing
```

Rubric:

```
[/1] Documentation

[/1] Part A correct

[/1] Part B
```