UrbanThings Algorithm Coding Exercise

Background

A hotel has **F** floors. It has installed a lift, which can take a maximum of **P** people at a time, up to a maximum weight of **W**.

Consider time as a series of discrete 'ticks'. In a single tick, the lift can do one of the following:

- Let any number of passengers off, then any number of passengers on.
- Move up or down by a single floor.

There is a queue of people waiting on the first floor. Assume that this is represented by an ordered set of their known weights (Array A) and destination floors (Array B).

You can assume that:

- The queue of people will enter the lift in a first-come-first-served basis until it is full.
- The lift will return to the first floor each time it is empty.
- Nobody ever requires to be taken to the first floor, since they are already there.
- There is no 'ground' floor.

For example, the following initial values:

```
A[] = {60, 80, 40}
B[] = {2, 3, 2}
F = 5
P = 2
W = 200
```

Will produce the following movement of the lift:

```
Tick   Lift Status
1      Loading at floor 1
2      Moving to floor 2
3      Unloading at floor 2
4      Moving to floor 3
5      Unloading at floor 3
6      Moving to floor 2
7      Moving to floor 1
8      Loading at floor 1
9      Moving to floor 2
10     Unloading at floor 2
11     Moving to floor 1
12     Completed
```

Given the above values, it would therefore take 12 ticks (include the final tick) for the lift to complete operations.

Task 1

Write a solution, in Swift (iOS roles) or Java or Kotlin (Android roles), to calculate how many ticks will be required for the lift to serve all the people in the queue then return to the first floor.

Your code should contain a method with the following signature:

```
int calculateLiftTicks(int[] A, int[] B, int F, int P, int W)
```

The arguments can be renamed as you wish. You can use as many methods and classes as you like.

You can use any IDE you wish. For Android it can be a pure JVM implementation or use Android SDK.

Deliver your solution class(es) **and also the unit tests that you use to prove that the implementation is correct**.

Task 2

The hotel has gained several more lifts.

Extend your code so that it performs the same calculation with multiple lifts. How many ticks will be required for the lifts to serve all the people in the queue then return to the first floor?

You may assume:

If more than one lift is at the first floor, people from the queue fill up each lift entirely before moving onto the next.

Your code should contain a new method with the following signature:

```
int calculateLiftTicks(int[] A, int[] B, int F, int P, int W, int numberOfLifts)
```

Task 3

The hotel designates some of the lifts as 'express lifts'. An express lift:

- Only stops at even numbered floors.
- Takes a single tick to travel up/down two floors.

Extend your code to deal with this. Assume that:

- Any passenger requiring an even numbered floor will always take an express lift, even if it means waiting.
- Any passenger requiring an odd numbered floor will never take an express lift, even if it means waiting.

Your code should contain a new method with the following signature.

```
int calculateLiftTicks(int[] A, int[] B, int F, int P, int W, int numberOfNormalLi
fts, int numberOfExpressLifts)
```

Submission instructions

Please provide a link by email to mark.woollard@urbanthings.co to a github repo with your solutions to the above tasks. The repo should have three branches `task1`, `task2` and `task3` with each containing the solution to each task. There should also be a `README.md` with any instructions needed to build and run each of the solutions.