(a) Give an algorithm in pseudocode (only pseudocode!) to find a stable assignment that is location optimal. Hint: it should be very similar to the Gale-Shapley algorithm, with locations taking the role of the men, and sales employees of the women.

```
Initialize Queue and add all locations into it
    While Queue is not Empty
        Remove an element (store #)
        Get the best employee preference that hasn't been used
        If employee is free & store has slots left to accept employees
            Match Employee and Store
            Decrease Store slot
        Else compare potential employee with matched employee'
            Check with Employee to see which store they prefer
            If employee prefers store over store',
                Unmatch employee'
                Match employee with Store
        Reenter store in Queue if store slots are available
```

(b) Using m as number of locations and n as number of sales employees, give the runtime complexity of your algorithm in (a) in Big O notation and explain why. Note: Full credit will be given to solutions that have a complexity of O(mn).

With this implementation, you do get a runtime of O(mn). By using the GS Algorithm for stable marriage problem, we propose in constant time and get an immediate answer by the employee. Depending on said answer and the results, we add it back to the queue. This allows us to only check the free locations and not repeat ones that will have no change in effect. At worst case scenario, you do have to revisit every location and employee, given O(mn).

(c) Give an algorithm in pseudocode (only pseudocode!) to find a stable assignment that is employee optimal Hint: it should be very similar to the Gale-Shapley algorithm, with employees taking the role of the men, and locations of the women.

```
Initialize a Queue and add all Employees as free
    While Queue is not Empty
        Remove an element (employee)
        Get their best store preference that hasn't been used
        If employee is free & store has slots left to accept employees
            Match Employee and Store
            Decrease Store slot
        Else compare employee with employee'
            Check with store to see which Employee it prefers based on its preference
            If store prefers Employee over Employee',
```

Unmatch employee' and put into queue if it still has options
Match employee with Store

(d) Give the runtime complexity of your algorithm in (c) in Big O notation and explain why.
Note : Try to make your algorithm as efficient as you can, but you may get full
credit even if it is not O(mn) as long as you clearly explain your running time
and the diffculty of optimizing it further.

Based on the GS Algorithm, the complexity would be O(mn). The algorithm provided above would keep a constant queue of free employees, and only adding when an unmatching occurs. Integrated within the code is that the first couple would most likely get their first choice of store location. If there were to be a conflict between an employee choosing a store with no more available spots, the store is allowed to choose its higher preference. If the store switches us, the employee that got unmatched would then be added to the queue, however, If not, then the employee goes to its next choice until he finds a store or no longer has any preferences given an uneven assignment of mn. In conclusion, the constant queue with the immediate accept/reject proposal creates the most efficient approach to this problem. Worst case scenario, they would have to revisit each m and n. Thus, O(mn) would be runtime complexity.