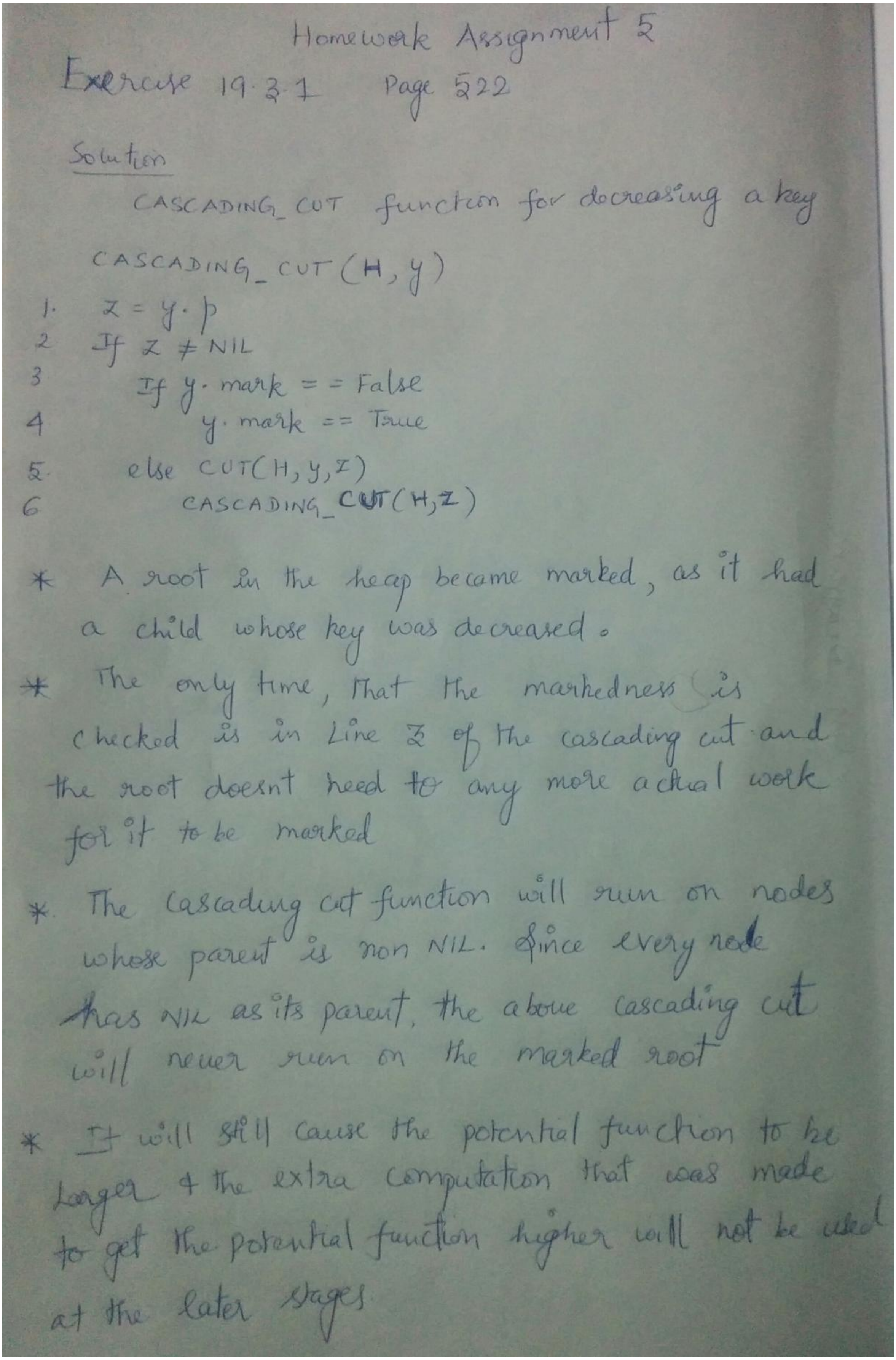
HOMEWORK ASSIGNMENT 5

1. Exercise 19.3.1 (Page 522)



Homework Assignment 5

Exercise 19.3.1 Page 522

Solution

CASCADING_CUT function for decreasing a key

```
CASCADING_CUT(H, y)
1.  z = y.p
2   If z ≠ NIL
3       If y.mark == False
4           y.mark == True
5.      else CUT(H, y, z)
6           CASCADING_CUT(H, z)
```

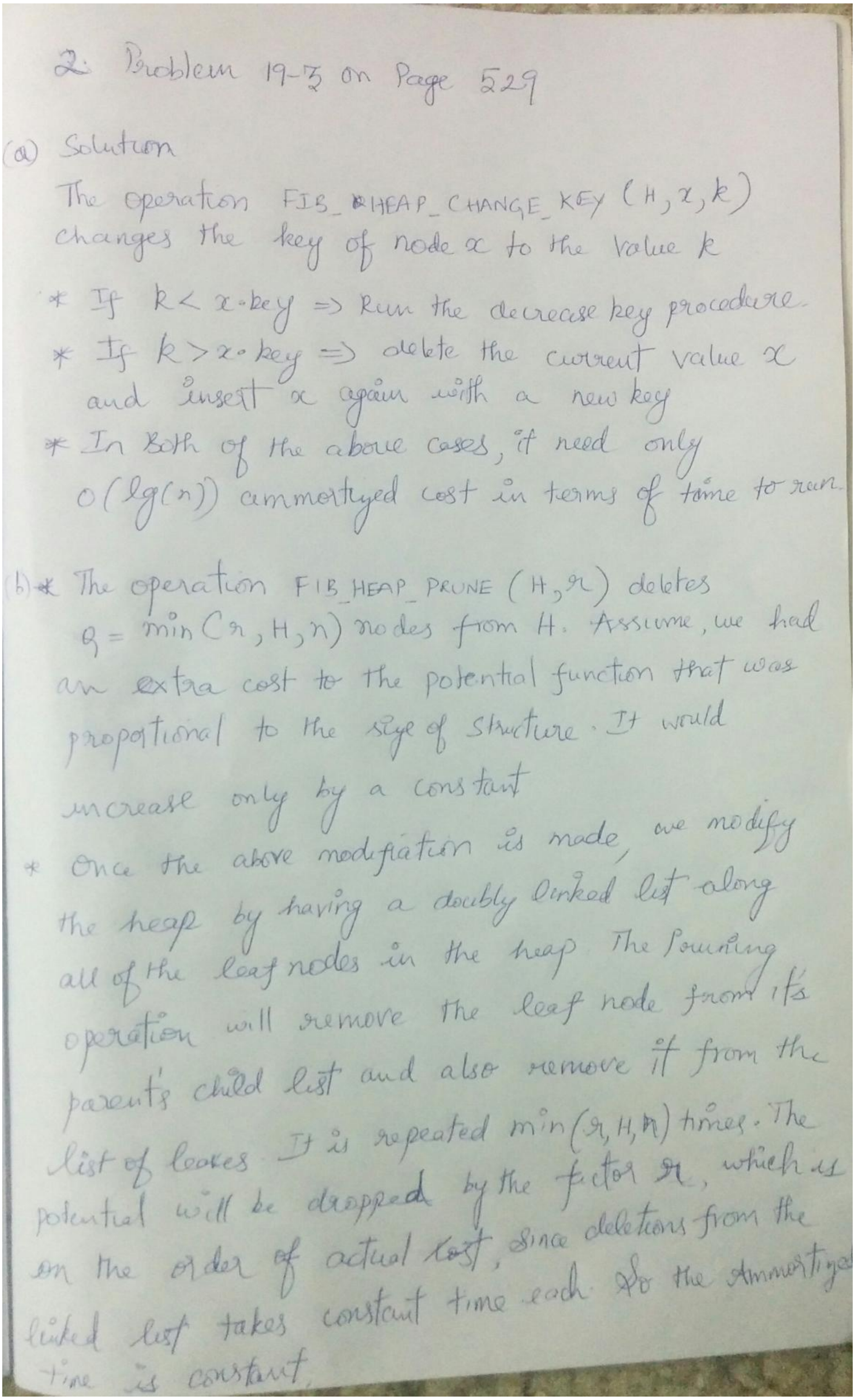
* A root in the heap became marked, as it had a child whose key was decreased.

* The only time, that the markedness is checked is in Line 3 of the cascading cut and the root doesnt need to any more actual work for it to be marked

* The cascading cut function will run on nodes whose parent is non NIL. Since every node has NIL as its parent, the above cascading cut will never run on the marked root

* It will still cause the potential function to be Larger & the extra computation that was made to get the potential function higher will not be used at the later stages.

2. Problem 19-3 (Page 529)



2: Problem 19-3 on Page 529

(a) Solution

The operation FIB_HEAP_CHANGE_KEY (H, x, k) changes the key of node x to the value k

- * If $k < x.key$ => Run the decrease key procedure
- * If $k > x.key$ => delete the current value x and insert x again with a new key
- * In Both of the above cases, it need only $O(lg(n))$ ammortyed cost in terms of time to run.

(b) * The operation FIB_HEAP_PRUNE (H, r) deletes $q = min(r, H, n)$ nodes from H. Assume, we had an extra cost to the potential function that was propotional to the size of structure. It would increase only by a constant

* Once the above modifiation is made, we modify the heap by having a doubly linked list along all of the leaf nodes in the heap. The Pouning operation will remove the leaf node from it's parent's child list and also remove it from the list of leaves. It is repeated $min(r, H, n)$ times. The potential will be dropped by the factor r, which is on the order of actual cost, since deletions from the linked list takes constant time each. So the Ammortized time is constant.