(a)
```
    ALGORITHM(L)                    //L is the list of processing times
1   n = |L|                         //get length of list input L
2   R = []                          //output
3   Lcopy = deepcopy(L)
4   sort(Lcopy)                     //sort in ascending order
5   for i = 0 to n − 1              //for each processing time P in Lcopy, find the index of
6       P = Lcopy[i]                //the first occurrence of P in L
7       for j = 0 to n − 1
8           if L[j] = P
9               L[j] = −1           //found an occurrence, set the occurrence to −1 so we don't
10              R.append(j+1)       //return this index again in case there are duplicates of
11              break               //this occurrence in L
12  return R                        //no elements can be −1 in L originally since t_1,...,t_n ∈ ℕ
```

(b) A partial solution for my algorithm is a list **K** that contains the indices of processing time(s) sorted such that the average completion time of executing all process(es) in the order specified by **K** is minimized.

Partial solutions: (Let R_n be the partial solution constructed by the algorithm after iteration n)

n = 0 $R_0 = []$
n = 1 $R_1 = [1]$
n = 2 $R_2 = [1, 4]$
n = 3 $R_3 = [1, 4, 3]$
n = 4 $R_4 = [1, 4, 3, 2]$

(c) Let R_0, R_1, ..., R_n be the **R** list at the end of each iteration of the loop on line 5. For any optimum solution OPT, we say OPT **extends** R_i if

- for $i > 0,\ i \in \mathbb{Z},\ OPT[k] = R_i[k]$ for $k = 0, ..., m$ where $m = |R_i| - 1$
- for $i = 0,\ R_i = []$

A partial solution, R_i is said to be promising iff $\exists OPT$ that extends R_i

(d) **Proof:** R_i is promising, for all i where $i = 0, ..., |L| - 1$ by induction on i (# of iterations)

(e) Lcopy contains $t_1, ..., t_n$ in a sorted ascending order. Let $T_1, ..., T_n$ be the elements of Lcopy.
Let K_1 be the index of T_1 in L, . . . , K_n be the index of T_n in L. The values of $K_1, ..., K_n$ are unique, for example, if some $T_i = T_j$ then there will be at least 2 occurrences of the value **G**, (G = $T_i = T_j$) in L. K_i will be the index of one occurrence of **G** in L and K_j will be the index of another occurrence of **G** in L

For each iteration, there is only one case:

- $R_{i+1} = R_i$ append K_{i+1}

(f) Given that

- $R_{i+1} = R_i$ append K_{i+1}

Does R_{i+1} extend OPT using our definition of **extend** in (c)?

There are two subcases:
Case 1: Yes, R_{i+1} extends OPT by our definition of **extend** therefore R_{i+1} is promising.
Case 2: No, R_{i+1} does not extend OPT by our definition of **extend**.

(g) Every R_i is promising, in particular, R_n is promising.
Therefore, ∃OPT such that:

- if n = 0, $R_n = []$
- if $n > 0,\ n \in \mathbb{Z},\ OPT[k] = R_n[k]$ for $k = 0, ..., m$ where $m = |R_n| - 1$