Worth: 2% **Due:** By 8:59pm on Tuesday 13 January

Remember to write your *full name* and *student number* prominently on your submission.

---

*Please read and understand the policy on Collaboration given on the Course Information Sheet. Then, to protect yourself, list on the front of your submission **every** source of information you used to complete this homework (other than your own lecture and tutorial notes). For example, indicate clearly the **name** of every student with whom you had discussions, the **title and sections** of every textbook you consulted (including the course textbook), the **source** of every web document you used (including documents from the course webpage), etc.*

*For each question, please write up detailed answers carefully. Make sure that you use notation and terminology correctly, and that you explain and justify what you are doing. Marks **will** be deducted for incorrect or ambiguous use of notation and terminology, and for making incorrect, unjustified, ambiguous, or vague claims in your solutions.*

Let $t_1, \ldots, t_n \in \mathbb{N}$ be *processing times* for $n > 0$ processes (for example, $t_1, t_2, t_3, t_4 = 2, 5, 3, 2$). Any permutation π of $[1, \ldots, n]$ specifies an order in which to execute the processes (for example, $\pi = [3, 2, 4, 1]$ specifies that process number 3 executes first, followed by process number 2, then process number 4, and finally process number 1).

For any permutation π and any index i, the <u>completion time</u> C_i is equal to the sum of the processing times of the first i processes, in the order given by π: $C_i = \sum_{j=1}^{i} t_{\pi[j]} = t_{\pi[1]} + \cdots + t_{\pi[i]}$ (for the example above, $C_1 = t_{\pi[1]} = t_3 = 3$, $C_2 = t_{\pi[1]} + t_{\pi[2]} = t_3 + t_2 = 3 + 5 = 8$, $C_3 = t_{\pi[1]} + t_{\pi[2]} + t_{\pi[3]} = t_3 + t_2 + t_4 = 3 + 5 + 2 = 10$, and $C_4 = t_{\pi[1]} + t_{\pi[2]} + t_{\pi[3]} + t_{\pi[4]} = t_3 + t_2 + t_4 + t_1 = 3 + 5 + 2 + 2 = 12$).

Given $t_1, \ldots, t_n$ as input, your task is to return a permutation π or $[1, \ldots, n]$ that *minimizes* the average completion time $(C_1 + \cdots + C_n)/n$.

(a) Write a greedy algorithm to solve this problem. For full marks, you must write your algorithm in pseudocode, following the conventions used in the textbook. *Please do **not** write your algorithm in Python, or C, or Java, or Haskell, or any other particular programming language.*

For this problem, it can be tempting to give a one line pseudocode algorithm. But it will be much easier to answer the rest of the questions if you take the time to write your algorithm as a loop that constructs its solution one item at a time.

(b) Define precisely the notion of "partial solution" for your algorithm from part (a). List every partial solution generated by your algorithm on the example input given in the statement of the question.

(c) Define precisely what it means for a partial solution to be "promising." Remember: part of your definition must include a precise description of what it means for some optimum solution to "extend" a partial solution. (There is **no** single, fixed definition for these notions that applies to every greedy algorithm, so they must be re-defined for each algorithm.)

(d) In class, we proved a statement by induction to help us conclude that the algorithm always produced an optimum solution—the statement was a particular loop invariant for the algorithm.

Write down this statement for your algorithm from part (a), clearly and precisely. Then state exactly what quantity you would do induction on to prove your statement.

(e) Consider the inductive proof of your statement from part (d). During the inductive step, the proof usually contains cases based on the possible choices made by the greedy algorithm.

What cases (if any) would be contained in the proof for your algorithm?

(f) For each case in the proof of the inductive step, the proof of correctness typically contains sub-cases based on the ways in which the optimum solution given by the induction hypothesis may or may not differ from the next partial greedy solution.

What sub-cases (if any) would be contained in the proof of each case for your algorithm?

(g) Finally, suppose that the loop invariant you stated in part (d) holds. Use this fact to prove that your algorithm from part (a) always produces an optimum solution.