NOTE: This file contains sample solutions to the quiz together with the marking scheme and comments for each question. Please read the solutions and the marking schemes and comments carefully. Make sure that you understand why the solutions given here are correct, that you understand the mistakes that you made (if any), and that you understand *why* your mistakes were mistakes.

Remember that although you may not agree completely with the marking scheme given here, it was followed the same way for all students. We will remark your quiz only if you clearly demonstrate that the marking scheme was not followed correctly.

For all remarking requests, please submit your request **in writing** directly to your instructor. For all other questions, please don't hesitate to ask your instructor during office hours or by e-mail.

GENERAL MARKING SCHEME:

- ***A****: All Correct*, except maybe for very few minor errors.
- ***B****: Mostly Correct*, but with a few serious errors, or many small errors.
- ***C****: Mostly Incorrect*, but with a few important elements, or many small elements, done correctly.
- ***10%****: Completely Blank*, or clearly crossed out.
- ***D****: All Incorrect*, except maybe for very few minor elements done correctly.

MARKER'S COMMENTS:

- **common error**: Unfortunately, few students managed to write a correct argument for question 2.

WARNING! This quiz involves a *lot* of reading, but only a *little* bit of writing.
Suppose you want to transfer a number of songs to CDs, using as few CDs as possible (the final order of the songs on the CDs does not matter). Formally, we define the problem as follows.

Input: A list of positive integer song *sizes* $[s_1, s_2, \ldots, s_n]$, and a positive integer *capacity* C.

Output: A partition of $[1, 2, \ldots, n]$ into sublists $L_1, L_2, \ldots, L_k$ (*i.e.*, each i belongs to exactly one L_j) such that k is **as small as possible** and no CD uses more than capacity C ($\forall j, \sum_{i \in L_j} s_i \leqslant C$).

This problem is *NP*-hard, but consider the following "First-Fit" approximation algorithm.

```
k ← 0
for i = 1, 2, ..., n:   # for each song
    # Put song i on the first CD L_j on which it fits.
    for j = 1, 2, ..., k:
        if s_i + ℓ_j ⩽ C:   # ℓ_j is the total size of the songs already on CD L_j
            L_j.append(i)
            ℓ_j ← ℓ_j + s_i
            continue with the next value of i
    # We get here only if s_i does not fit on any existing L_j; in this case, create a new CD.
    k ← k + 1
    L_k = [i]
    ℓ_k = s_i
```

1. For any input $s_1, s_2, \ldots, s_n$, let k be the number of CDs generated by this algorithm, and k^* be the minimum number of CDs required to store all the songs. Give a precise definition of what it means for the algorithm to have approximation ratio r.

$$k \leqslant r \cdot k^*$$

2. Use the following facts to show that the algorithm has an approximation ratio of at most 2.

- Every solution uses *at least* $\sum_{i=1}^{n} s_i / C$ CDs.

- The solution produced by the algorithm leaves *at most* one CD less than half full. In other words, the total size of the songs on any *two* CDs is greater than the capacity C.

HINT: Think about $L_1 + L_2 + \cdots + L_k$ and how it relates to the total size of all the songs $\sum_{i=1}^{n} s_i$.

Since every song is assigned to a CD, we conclude that $L_1 + L_2 + \cdots + L_k = s_1 + s_2 + \cdots + s_n$.
From the first fact, we conclude that $k^* \geqslant \sum_{i=1}^{n} s_i / C$.
From the second fact, we conclude that $L_1 + L_2 + \cdots + L_k > C(k/2)$.
This implies that $k < 2(L_1 + L_2 + \cdots + L_k)/C$

$$= 2(s_1 + s_2 + \cdots + s_n)/C$$
$$\leqslant 2k^*.$$