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: *Mostly well done.*

1. Recall the definition of the subset sum *decision* problem and consider the related subset sum *search* problem.

subset sum decision problem:

Input: A set of positive integers $S = \{x_1, x_2, \ldots, x_n\}$ and a positive integer target t.

Question: Is there some subset of S whose sum is exactly t?

subset sum search problem:

Input: A set of positive integers $S = \{x_1, x_2, \ldots, x_n\}$ and a positive integer target t.

Output: A subset of S whose sum is exactly t, or the special value NIL if there is no such subset.

Give a detailed argument to show that the subset sum problem is polytime self-reducible. Make sure to include a brief English description of the main idea of your algorithm, to justify that your algorithm is correct, and to analyze your algorithm's running time.

Suppose that SSD solves the decision problem in time $t(n)$. Then the following algorithm solves the search problem.

```
SSS(S, t):
    if not SSD(S, t):  return NIL
    for each x ∈ S:
        if SSD(S − {x}, t):
            S ← S − {x}
    return S
```

Runtime: $\text{SSS}(S, t)$ runs in time $\mathcal{O}(nt(n) + n^2)$: it makes a linear number of calls to SSD and creates a linear number of sets S, each one in worst-case linear time.

Correctness: The fact that S contains some subset whose sum is exactly t is a loop invariant: it is true at the beginning (because of the initial call to SSD) and it remains true after each iteration (because S is only changed when the resulting set still contains some subset with sum exactly t). So at the end of the loop, S contains a subset whose sum is exactly t.

However, the final value of S contains nothing else because every original value that is not needed to achieve sum t will be removed by the algorithm. Hence, the algorithm correctly solves the search problem.