Technical assessment test

Formal Vindications S.L.

This is an exercise intended to be done in Coq. Please read all the questions before you start answering them.

Your answer should come in a Coq source file called `ta-test-fv.v`. You can also provide a pdf with the natural language answers to the questions 3-5, but there's no need – Coq comments are enough. You can import anything of the Coq Standard Library, SSReflect and MathComp. Explanations attached to the code in the format of comments are very welcome.

If you can't finish some of the exercises, don't worry. You can write in a comment what are your difficulties to get to an end. We are interested in seeing what's your style and level, not only in Coq but also in your own reflections!

1. Write a function with signature

   ```
   add_greatest2 :  list nat -> nat
   ```

   (or `seq nat` if you choose MathComp) which, given a list with at least two elements, gives the sum of the two greatest elements (and, in case the given list has less than two elements, the result is left to your own choice). Of course, you can write any auxiliary functions you need.

2. State and prove a lemma `add_greatest2_correct` saying that, given a list `s` and two elements `a, b` of the list, the sum of `a` and `b` is smaller or equal than `add_greatest2 s`. Again, you can write any auxiliary lemmas you need.

3. Do you think that this lemma specifies the function entirely? In other words, is this lemma enough to consider the function verified? If your answer is yes, write a one-sentence explanation (as a comment). Otherwise, state the lemma or lemmas that you think would be enough (proving them is not necessary).

4. Assume that now you need to write a function with signature

   ```
   add_greatest :  nat -> list nat -> nat
   ```

   which receives a natural number `n` and a list `s` and computes the sum of the `n` greatest numbers of `s`. Would you change your algorithm? Describe what you would do (no need to write the code). Do you think the proof of correctness would be much harder in this case? Write your thoughts on the topic.

5. Now suppose you need to extract the code from Exercise 1 to OCaml. You know that the natural numbers on the list can range from 0 to 2^{30}, and the list itself can have at most 10^6 elements[1]. What would happen if you just did straight-forward extraction? What ideas would you have to avoid the execution issues derived from the extraction of natural numbers?

[1] The numbers are just an orientation, no need to do any math with them.