Exercice 1. Proving programs 1 Write a short C-function float arrayMax(const float * A, int size) which computes the maximal value stored in the array, i.e. such that for every array a of size n,

\[
\text{arrayMax}(A, n) = \max_{0 \leq i < n} A[i].
\]

A correct (hopefully) implementation is, assuming \(\max\emptyset = -\infty\) by convention,

```c
float arrayMax(const float * A, int size){
    float m = -std::numeric_limits<float>::infinity();
    for(int i = 0; i < size; i++) {
        if(A[i] < m) m = A[i];
    }
    return m;
}
```

As seen in class, even such a simple program may contain bugs in some variants written without care, for example forbidden access to \(A[0]\).

To prove that the implementation is correct is not easy to do in a formal way, because there is no formal model of C. However we can check that the property

\[
P(i): m = \max_{0 \leq j < i} A[j]
\]

is always true at step \(i\) of loop, in other words \(P(i)\) is an invariant of the loop.

When \(i = 0\), this is obvious because \(m = \max\emptyset\). Moreover if \(P(i)\) is true at some step \(i < n\) then \(P(i + 1)\) is true at step \(i + 1\), this is quite obvious. Thus \(P(n)\) is true i.e. the function indeed returns \(\max_{0 \leq i < n} A[i]\).

Exercice 2. Proving programs 2 Let bool isSorted(const float * A, int size) be a function which checks that the input is sorted in increasing order.

Write a formula which defines what the function \(\text{isSorted}\) is supposed to compute.

\[
\text{isSorted}(A, n) = \begin{cases} 
    \text{true} & \text{if } \forall 0 \leq i < j \leq n, A[i] \leq A[j] \\
    \text{false} & \text{otherwise.}
\end{cases}
\]
Provide an implementation;

```c
float isSorted(const float * A, int size){
    for(int i = 0; i < size-1; i++)
        if(A[i+1] > A[i]) return false;
    return true;
}
```

Prove that your implementation is correct. We use the invariant:

\[ P(i) : \forall 0 \leq j < k < i, A[j] \leq A[k] \]

which is true at the beginning of the \( i \)-th of the loop. Indeed, clearly \( P(0) \) is true (because the statement is void) and we can check that if \( P(i) \) is true then \( P(i+1) \) is true as well.

There are two kinds of execution. Either the algorithm exits the loop before step \( n \), because two consecutive elements are in the wrong order, in which case the array is not sorted and the result returned (false) is correct.

Or the algorithm performs all steps of the loop, in which case \( P(n) \) is true thus the array is sorted and the algorithm is correct.

**Exercice 3. Proving programs 3** Same exercise than previous for a function

```c
bool inArray(const float * A, int size, float x) which checks whether \( x \) appears in the array \( A \).
```

Similar answer, send email to me if case you need a detailed answer.