Coursework 2. Submission deadline 10/12/18

1 Fast exponentiation (20 marks)

In this task the input consists of two natural numbers a and n and the task is to compute a^n using multiplication only (that is, without the use of exponentiation function).

One way to do this is the following.

```
exp=1
for i in range (1,n+1):
  exp=exp*a
print(exp)
```

It is not hard to see that the above program performs n exponentiations. That is, its runtime is $O(n)$.

Write a program that performs the same task in $O(\log n)$ (that is, performs $O(\log n)$ exponentations). You need to accompany your solution with a short explanation as to why you believe your solution is be correct. **Marks will not be awarded with such an explanation!**

Remark 1. It is up to you whehter to present your algorithm in the recusrive or in the iterative form. However, it is important to note that, for this particular task, the recursive solution is way easier.

The iterative solution is quite non-trivial. For instance, in the one I have in mind, the binary representation of n is computed first and maintained in an array. That said, under the assumption that n is a power of 2, the iterative solution becomes easy as well.

Remark 2. If you find it difficult to solve the problem in full generality, try to solve under assumptionthat n is a power of 2. 10 marks will be awarded for a correct solution under such an assumption but **only on condition that the assumption is explicitly stated**.

2 Missing number problem with a sorted array (20 marks)

Consider the missing number problem (see slides for Lecture 1) under assumption that the input array is sorted. Write a program solving this problem in $O(\log n)$ under this assumption.

Hint: Try to use binary search. The only non-trivial point is to understand the condition that allows to discard half of the array at once.

3 Subarray problem

Consider the following task given two arrays A and B *without repetitions* (that is, no double occurrences of the same element). Tha task is to check whether each element of B is also an element of A without regard of the order.

For instance if $A = [1, 2, 3, 4]$ and $B = [2, 3, 1]$ then the answer is YES. If however $B = [1, 2, 5]$ then the answer is NO because 5 is not in A. Such a problem is easy to solve using the set operations in Python. **For this exercise please do not use these operations. Marks will not be awarded for solutuions using the Python operations with sets**

3.1 Iterative algorithm (20 points)

Design an $O(n^2)$ algorithm for the subarray problem.

Hint: use two nested loops. You need to think what to explore in the inner and in the outer loops.

3.2 Recursive algorithm (20 points)

Design a recursive algorithm (no use of loops) for the subarray problem.

3.3 Arrays sorted (20 points)

Assume that both A and B are sorted arrays and design an $O(n)$ algorithm under this assumption.

Hint: A possible solution is a simple modification of the $Merge$ procedure of $MergeSort$ algorithm.