Exercise 1: Hashing with Open Addressing  (10 Points)

We consider hash tables with open addressing and two different methods for collision resolution: **linear probing** and **double hashing**. Let $m$ be the size of the hash table where $m$ is prime. Let $h_1(x) := 53 \cdot x$ and $h_2(x) := 1 + (x \mod (m-1))$. We define the following hash functions for collision resolution according to the lecture:

- **linear probing**: $h_\ell(x,i) := (h_1(x) + i) \mod m$.
- **double hashing**: $h_d(x,i) := (h_1(x) + i \cdot h_2(x)) \mod m$.

(a) Implement a hash table with operations `insert` and `find` using the mentioned strategies for collision resolution. You may use the template `HashTable.py`.  

(b) Create a hash table of size $m > 1000$ ($m$ prime) and measure the average time for inserting $k$ keys for $k \in \{\lfloor \frac{m}{50} \rfloor | i = 1, \ldots, 49\}$ in four variations: Using linear probing / double hashing; inserting $k$ random keys / the set of keys $\{m \cdot i | i = 1, \ldots, k\}$. Create a plot showing the four different average runtimes. Discuss your results in `erfahrungen.txt`.  

Exercise 2: Application of Hashtables  (10 Points)

Consider the following algorithm:

```
Algorithm 1 algorithm
1: for i = 1 to n - 1 do
2:    for j = 0 to i - 1 do
3:        for k = 0 to n - 1 do
5:                return true
6: return false
```

> Input: Array A of length $n$ with integer entries

(a) Describe what `algorithm` computes and analyse its asymptotical runtime.  


(b) Describe a different algorithm $B$ for this problem (i.e., $B(A) = algorithm(A)$ for each input $A$) which uses hashing and takes time $O(n^2)$ (with proof).  

*Hint: You may assume that inserting and finding keys in a hash table needs $O(1)$ if $\alpha = O(1)$ (\(\alpha\) is the load of the table).*
(c) Describe another algorithm for this problem without using hashing which takes time $\mathcal{O}(n^2 \log n)$ (with proof). (4 Points)