- Hash table with Quadratic Probing.

Implement a simple hashtable using quadratic probing for collision resolution. Both the keys and values are integers, assuming greater than 0. The initial table size m should be 11 (it is too small for a real hashtable, we just use it for the purpose of this homework). Let n be the number of items in the table. When $n \geq m/2$, use the technique of dynamic arrays to enlarge the table. You want to approximately double the table size but keep to the primes. The next table size m will be 23.

You should use key%m as the hash function.

Let b be the hash value modulo m. If bucket b is occupied, you probe $(b + 1^2) \% m$, $(b + 2^2) \% m$, $(b + 3^2) \% m$, $\cdots$, $(b + (m/2)^2) \% m$, and stop as soon as you find an empty bucket.

As long as n is kept less than m/2, you will find an empty bucket by the end of the probing.

You should at least implement the following functions:

(a) void put(int key, int value): insert key-value pair to the hashtable. Insert key to the key array, value to the value array based on the hash function.

(b) int get(int key): get the value of the key

(c) boolean contains(int key): return true if the hashtable contains the key

(d) void remove(int key): remove the key-value pair

(e) void rehashing(): this method is called automatically when $n \geq m/2$. You should enlarge the table and use findPrime($2 * m$) to get the new table size. You need to compute new hash index for every key stored in the existing hash table.

(f) int findPrime(int x): find the next (the smallest) prime bigger than x. For example, findPrime(8) = 11, findPrime(22) = 2