Question - 1

Hash - 1

SCORE: **5 points**

Which of the following statement(s) is TRUE?

1. A hash function takes a message of arbitrary length and generates a fixed length code.
2. A hash function takes a message of fixed length and generates a code of variable length.
3. A hash function may give the same hash value for distinct messages.

○ I only

○ II and III only

● I and III only

○ II only

Question - 2

Hash - 2

SCORE: **5 points**

Suppose we are using Hash(k) = 3 * k % 13, and an array of size 13 for our Hash Table, what's the result after inserting the numbers below into the hash table if we use linear probing? (where * represents that there is no value at that index). Note that the array indices are, as usual, 0 thru 12.
Numbers in order: 22 -> 40 -> 36 -> 55 -> 24 -> 27 -> 28

○ * 22 * 40 36 27 * 24 * 55 28 * *

○ 22 * 40 36 27 28 24 * 55 * * * *

○ 22 * 27 36 28 * 24 * 55 * * * *

○ * 22 * 40 27 36 * 24 * 55 * * *

○ * 22 * 27 36 28 * 24 * * * * *

● * 22 * 40 36 27 28 24 * 55 * * *

Question - 3

Hash - 3

SCORE: **5 points**

In simple uniform hashing, where there are *n* keys, what is the order of complexity for search?

○ O(n)

? Help

○ O(logn)

○ O(nlogn)

● O(1)

Question - 4

Hash - 4

SCORE: **5 points**

Suppose we have a mutable class *X* which contains two attributes: *name* and *ID*. We manually override the *hashCode* function with our own implementation that returns a value based on both attributes, for example, *name.hashCode() + ID*31*. Note that this is just an example, not the definition of any particular hash function.

First, we create a new instance *X x = new X("INFO6205", 27)*. We then add this instance (*x*) into an empty HashSet (*s*), i.e. *s.put(x)*. Next, we modify the *ID* of this instance, e.g. *x.setID(42)*.

Finally, we invoke the *contains* function, i.e. *s.contains(x)*. What result should we expect?

○ True

○ False

○ Runtime Exception

○ Null

● Probably false but possibly true.

Question - 5

Hash - 5

SCORE: **5 points**

The advantages of separate chaining as an implementation of a hash table over the linear probing (open addressing) scheme include:

○ Space used is less

● Deletion is easier

● As more keys are added, performance degrades gracefully rather than suddenly blowing up.

○ None of the above

Question - 6

Coding

SCORE: **30 points**

Please implement the *put* and *get* methods for a linear probing hash table.