Python 3.6

Assume that Python dictionaries (`dict`), up to version 3.6, were implemented with linear probing as taught in IN2010. (This is a slight modification of the truth).

In December 2012, Raymond Hettinger proposed a change to the underlying representation of dictionaries. In the proposal, Hettinger described that the current representation was unnecessarily inefficient.

It organized the data in an array `entries`, where each element has space for a key k and a value v. The array has size N and a lot of empty space. Hettinger proposed reorganizing the data. The key-value pairs should instead be stored in a tightly packed, dynamic array `entries` with n elements, where n is the number of key-value pairs. The elements in `entries` can be referenced from another array `indices` with N elements (which still has a lot of empty space).

For example, the following dictionary:

```
d = {}
d["timmy"] = "red"
d["barry"] = "green"
d["guido"] = "blue"
```

was previously represented like this:

```
entries = [(_, _),
           ("barry", "green"),
           (_, _),
           (_, _),
           (_, _),
           ("timmy", "red"),
           (_, _),
           ("guido", "blue")]
```

where `(_, _)` indicates an empty space. The proposal is to represent it like this:

```
indices = [_, 1, _, _, _, 0, _, 2]
entries = [("timmy", "red"), ("barry", "green"), ("guido", "blue")]
```

The above proposal led to the implementation of dictionaries in Python being changed from version 3.6.

(a) Briefly explain, in natural language, how the algorithms for lookup and insertion with linear probing need to be changed to account for the new structure.

(b) What is a rehash? What causes a rehash?

(c) How will a rehash proceed in the new structure?

(d) What are the advantages of the new structure?