You may work in pairs for this assignment. If you choose to work with a partner, make sure only one of you submits a solution, and you paste a copy of the Partners Template that contains the names and PIDs of both students at the beginning of the file you submit.

You will submit your solution to this assignment to the Curator System (as `HW02`). Your solution must be either a plain text file (e.g., NotePad++) or a typed MS Word document; submissions in other formats will not be graded.

Partial credit will only be given if you show relevant work.

---

1. Consider a hash table consisting of $M = 11$ slots, and suppose nonnegative integer key values are hashed into the table using the hash function `h1()`:

```
public static int h1( int key ) {

   int x = (key + 11) * (key + 13);
   x = x << 4;
   x = x + key;
   return x;
}
```

a) [27 points] Suppose that collisions are resolved by using quadratic probing, with the probe function:

$$\left(k^2 + k\right)/2$$

The integer key values listed below are to be inserted, in the order given. Show the home slot (the slot to which the key hashes, before any probing), the probe sequence (if any) for each key, and the final contents of the hash table after the following key values have been inserted in the given order:

Key Value	Home Slot	Probe Sequence
43		
23		
15		
27		
31		
14		
37		
21		
29		

Final Hash Table:

Slot	0	1	2	3	4	5	6	7	8	9	10
Contents											

b) **[27 points]** Suppose that collisions are resolved by using double hashing (see the course notes), with the secondary hash function `Reverse(key)`, which reverses the digits of the key and returns that value; for example, `G(7823) = Reverse(7823) = 3287`.

The integer key values listed below are to be inserted, in the order given. Show the home slot (the slot to which the key hashes, before any probing), the probe sequence (if any) for each key, and the final contents of the hash table after the following key values have been inserted in the given order:

Key Value	Home Slot	Probe Sequence
43		
23		
15		
27		
31		
14		
37		
21		
29		

Final Hash Table:

Slot	0	1	2	3	4	5	6	7	8	9	10
Contents											

2. A hash table is being implemented, using double hashing to resolve collisions. We already have our primary hash function, `H()`, to hash the key value, and we have decided on a table size of 997, expecting to store no more than about 500 elements in the table.

Haskell Hoo proposes to make double hashing more efficient by adopting the following scheme. Rather than use a second hash function, which would require another function evaluation, Haskell proposes that we just calculate the step distance for probing this way:

```
step_distance = 997 - home_slot_index
```

a) **[12 points]** One potential problem with double hashing is that it might only access a few table slots before repeating itself. Would Haskell's scheme avoid **that difficulty**? Explain. (There may be other difficulties with this idea, but we are not considering them here.)

b) **[12 points]** In any case, no matter what conclusions you reached in part a), there is a fundamental, **fatal** flaw in Haskell's scheme for double hashing that has nothing to do with the issue considered in part b). Explain.

As a second alternative, Haskell proposes computing the probe step distance for double hashing by reusing the result from the original hash function, instead of having to evaluate a second function. Specifically, he proposes that we compute the step distance for probing this way:

```
step_distance = 17 + H(key)
```

c) [12 points] Would Haskell's second scheme avoid **the difficulty considered in part a)**? Explain. (There may be other difficulties with this idea, but we are not considering them here.)

d) [10 points] Would Haskell's second scheme avoid **the difficulty you identified in part b)**?