HW3
Name: Xiang Gu
EID: xg2847

Part A:

14.7.1
Speed Bitmap Index:

Speed	Original Bit-vector	Compressed Bit-vector
1.42	001000000000	1010
1.86	000000000010	11101010
2.00	000000001000	11101000
2.10	010000000000	01
2.20	000000110000	11011000
2.66	100000000000	00
2.80	000100000101	101111010101
3.20	000011000000	11010000

Ram bitmap index:

ram	Original bit-vector	Compressed bit-vector
512	011010000000	010001
1024	100101101001	0010100100011010
2048	000000010110	1101110100

Hd bitmap index:

hd	Original bit-vector	Compressed bit-vector
80	001000000000	1010
160	000000000011	1110101000
200	000000100000	110110
250	110110011000	00000100101000
300	000000000100	11101001

320	000001000000	110101

14.7.3

m * (1,000,000 / 8) bytes

Explanation: each bit-vector (for each value) has 1 million / 8 bytes. Hence the bitmap index has in total m * (1 million / 8) bytes.

Part B:

1.
(a). There is **no** false positive after adding a through y;
(b). **No**, it does not depend on the size of the key.

2.
The general formula for the probability of having a false positive after inserting n keys in a hash table of length m bits with k hash function is (1 - (1 - 1/m) ** kn) ** k
(a). Plug in m = 50, k = 3, n = 1, Pr(false positive after inserting one key) \approx **0.0002**
(b). Plug in m = 50, k = 3, n = 20, Pr(false positive after inserting twenty keys) \approx **0.3466**
(c). Definitely **not** there (voted on Piazza poll)

3.
The optimal k* = m/n * ln2. After plugging in m = 50 and k =3, we can write down the inequality 2 < 50/n * ln2 <= 3, from which we can solve for n: 11.55 <= n < 17.328.
Therefore, the possible number of keys the requirements statement stated to be stored are **{12, 13, 14, 15, 16, 17}**, for this k = 3 to be optimal for this particular length of the hash table (i.e. m = 50).

Part C:

5.1.1

Speed (as a set)	Speed (as a bag)
2.66	2.66
2.10	2.10
1.42	1.42
2.80	2.80
3.20	3.20

2.20	3.20
2.00	2.20
1.86	2.20
3.06	2.00
	2.80
	1.86
	2.80
	3.06
Average = 2.3667	Average = 2.4846

5.1.2:

hd (as a set)	hd (as a bag)
250	250
80	250
320	80
200	250
300	250
160	320
	200
	250
	250
	300
	160
	160
	80
Average = 218.33	Average = 215.38

16.2.2:
(b).
set difference:
Consider the following two relations R(a,b) and S(a,b):

R = {(1,3), (1,3)}, S = \Phi (aka empty relation)

\pi_a (R - S) = {(1)}   !=   \pi_a(R) - \pi_a(S) = {(1), (1)}

bag difference:
Consider the following two relations R(a,b) and S(a,b):

R = {(1,3), (1,2)}, S = {(1,3), (1,4)}

\pi_a(R -B S) = {(1)}   !=   \pi_a(R) -B \pi_a(S) = \Phi

(c).
Consider the following relation R(a,b):

R = {(1,3), (1,4)}

\delta(\pi_a(R)) = {(1)}   !=   \pi_a(\delta(R)) = {(1), (1)}

2.
i.

```
p3=# select * from r where joinKey1 = 101;
 theprimarykey |  name  | joinkey1
---------------+--------+----------
             1 | Andrea |      101
(1 row)
```

ii.

```
p3=# select * from r where joinKey1 != 101;
 theprimarykey | name | joinkey1
---------------+------+----------
             5 | John |      106
(1 row)
```

iii.

```
p3=# select * from r where joinKey1 IS NULL;
 theprimarykey | name  | joinkey1
---------------+-------+----------
             2 | David |
             3 | David |
             4 | Dan   |
(3 rows)
```

iv.

```
p3=# select name, joinKey1 from r;
  name  | joinkey1
--------+----------
 Andrea |      101
 David  |
 David  |
 Dan    |
 John   |      106
(5 rows)
```

v.

```
p3=# select joinKey1 from r;
 joinkey1
----------
      101



      106
(5 rows)
```

vi.

```
p3=# select * from r, s where joinKey1 = joinKey2;
 theprimarykey |  name  | joinkey1 | theprimarykey | romannumeral | joinkey2
---------------+--------+----------+---------------+--------------+----------
             1 | Andrea |      101 |             6 | V            |      101
(1 row)
```

vii.

```
p3=# select * from r, s where joinKey1 != joinKey2;
 theprimarykey |  name  | joinkey1 | theprimarykey | romannumeral | joinkey2
---------------+--------+----------+---------------+--------------+----------
             5 | John   |      106 |             6 | V            |      101
             1 | Andrea |      101 |             8 | L            |      105
             5 | John   |      106 |             8 | L            |      105
(3 rows)
```

viii.

```
p3=# select * from r
p3-# full outer join s on joinKey1 = joinKey2;
 theprimarykey |  name  | joinkey1 | theprimarykey | romannumeral | joinkey2
---------------+--------+----------+---------------+--------------+----------
             1 | Andrea |      101 |             6 | V            |      101
               |        |          |             8 | L            |      105
               |        |          |             7 | X            |
             5 | John   |      106 |               |              |
             2 | David  |          |               |              |
             3 | David  |          |               |              |
             4 | Dan    |          |               |              |
(7 rows)
```

ix.

```
p3=# select * from r
p3-# left outer join s on joinKey1 = joinKey2;
 theprimarykey |  name  | joinkey1 | theprimarykey | romannumeral | joinkey2
---------------+--------+----------+---------------+--------------+----------
             1 | Andrea |      101 |             6 | V            |      101
             5 | John   |      106 |               |              |
             2 | David  |          |               |              |
             3 | David  |          |               |              |
             4 | Dan    |          |               |              |
(5 rows)
```

x.

```
p3=# select * from r where joinKey1 in (select joinKey2 from s);
 theprimarykey |  name  | joinkey1
---------------+--------+----------
             1 | Andrea |      101
(1 row)
```

xi.

```
p3=# (select * from r)
p3-# except
p3-# (select * from r where joinKey1 in (select joinKey2 from s));
 theprimarykey | name  | joinkey1
---------------+-------+----------
             5 | John  |      106
             4 | Dan   |
             2 | David |
             3 | David |
(4 rows)
```

3.
Three ways to write query vi: (theta-join)

- select * from r,s where joinKey1 = joinKey2;
- select * from r inner join s on joinKey1 = joinKey2;
- with r1 as (select * from r where joinKey1 in (select joinKey2 from s)),
      s1 as (select * from s where joinKey2 in (select joinKey1 from r))
  select * from r1, s1;

Three ways to write query vii: (theta-join)

- select * from r,s where joinKey1 != joinKey2;
- select * from r inner join s on joinKey1 != joinKey2;
- (select * from r,s where joinKey1 is not null and joinKey2 is not null)
  except
  (select * from r,s where joinKey1 = joinKey2);

4.
Two ways to write query viii: (full outer join)

- select * from r full outer join s on joinKey1 = joinKey2;
- (select * from r left outer join s on joinKey1 = joinKey2)
  union
  (select * from r right outer join s on joinKey1 = joinKey2);

Two ways to write query ix: (left outer join)

- select * from r left outer join s on joinKey1 = joinKey2;
- (select * from r full outer join s on joinKey1 = joinKey2)
  except
  (select * from r right outer join s on joinKey1 = joinKey2);

Two ways to write query x: (left semijoin)

- select * from r where joinKey1 in (select joinKey2 from s);
- (select * from r)
  except

(select * from r where joinKey1 not in (select joinKey2 from s));

Two ways to write query xi: (left anti semijoin)

- select * from r where joinKey1 not in (select joinKey2 from s);
- (select * from r)
  except
  (select * from r where joinKey1 in (select joinKey2 from s));