Solution 1:

a) (5 pts.) $O(n^2)$

b) (5 pts.) $O(n^4)$
Solution 2:

a) (5 points)
For the main diagonal, i = j
For the anti-diagonal i+j = n-1 (or n+1 if indexing starts at 1)

b) (10 points)
One can use the first n elements in the array for the diagonal, and then put in the anti-diagonal.

One important point is to make sure that when n is odd, one element would be in both diagonals, so one needs to care of that.

In our solution we will skip that element on the anti-diagonal if n is odd.

c) (10 points)
We need to make sure that both the above problem is solved and that the method works for both odd and even n.

We're assuming 0-based indexing.

```java
int get(int i, int j) {
    if (i == j) return elements[i];
    else if (i + j == n - 1) {
        if (i < j) return elements[n + i];
        else return elements[n + i - 1]
    }
    else
        return 0;
}
```
Solution 3:

a) (10 points)

Code 0 1 2 3 4 5

Key a b aa aab ba aaba

Compressed code: 0 2 1 3 0 1

b) (10 points)

Code 0 1 2 3 4 5

Key a b ab ba aa aba

Decompressed code: abaaabaab
Solution 4:
   a) (10 points)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>13</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>13</td>
<td>31</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>13</td>
<td>31</td>
<td>24</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>19</td>
<td>13</td>
<td>31</td>
<td>24</td>
<td>23</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>19</td>
<td>13</td>
<td>31</td>
<td>24</td>
<td>23</td>
<td>32</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Loading factor = 7/9

   b) (5 points!) The maximum number of buckets examined in an unsuccessful search is 6, when searching a key % 9 = 4

The average number of buckets examined in an unsuccessful search:
(1+3+2+1+6+5+4+3+2)/9=3

The maximum number of buckets examined in a successful search is 4, when searching 32

The average number of buckets examined in a successful search:
(1+2+1+2+1+3+4)/7=2

c) (5 points)

<table>
<thead>
<tr>
<th>0</th>
<th>19</th>
<th>31</th>
<th>23</th>
<th>24</th>
<th>32</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Solution 5: (25 points)

```c
void reverseOrder()
    ChainNode* p = firstNode;
    ChainNode* tmp = null, q = null;
    while (p != null){
        tmp = p;
        p = p->next;
        tmp->next = q;
        q = tmp;
    }
    firstNode = q;
```