1. Observe the following:

$$\sum_{i=0}^{r} \sigma_i 128^i \equiv \sum_{i=0}^{r} \sigma_i 1^i \ (mod\ 127)$$
$$\equiv \sum_{i=0}^{r} \sigma_i \ (mod\ 127)$$

Since strings containing the same characters will yield the same sum of σ_i values and addition is commutitive, we may conclude that any two strings that are permutations of each other will be hashed to the same value.

2a) Note that many possible solutions exist for this question. Let our hash function be h(x) = x (mod 10), m=10. Suppose our operations are Insert(1), Insert(11), Insert(21), Search(1). In T_1, the element containing 1 will be at the end of the linked list requiring 3 comparisons to find it. In T_2, this same element will be at position h(1) as expected, requiring only one comparison.

2b) There were many submissions referring to the expected number of comparisons derived in lecture. The assumptions about the probability space used between this question and that derivation are different, and that derivation didn't take into account the issue of deleted symbols. In any case, when given such a problem, it is best to derive a proof on your own using first principles. Below is one such example.

Consider the state of T_1 and T_2 after an arbitrary sequence of n INSERT and DELETE operations. Since T_1 and T_2 implement the same ADT, they have exactly the same items in them. Furthermore, since T_1 and T_2 have the same number of buckets and use the same hash function, each item hashes to the same bucket in both tables.

Let m be the number of buckets in T_1 (also T_2). Let p be the number of items in T_1 (also T_2). So $p \leq n$. For each $0 \leq i < m$, let L_i be the number of items in T_1 (also T_2) that hashes to bucket i. So $\sum_{i=0}^{m-1} L_i = p$.

For $0 \leq i < m$ and $1 \leq j \leq L_i$, let $x_{i,j}$ be the j-th item in the chain from bucket i of T_1, and let $y_{i,j}$ be the j-th item found in the probe sequence starting at bucket i that hashes to bucket i.(Note that $x_{i,j}$ and $y_{i,j}$ are not necessarily the same, but $x_{i,j}$ is defined exactly when $y_{i,j}$ is defined.)

For each table, the expected number of item comparisons equals the total number of item comparisons when we search for every item once, divided by the number of items in the table.

For T_1, the number of item comparisons to search for $x_{i,j}$ is j since we are searching for the jth item in a doubly linked list. Thus $E_1 = \frac{1}{p} \cdot \sum_{i=0}^{m-1} \sum_{j=1}^{L_i} j$

For T_2, the number of item comparisons to search for $y_{i,j}$ is at least j since we are searching for the jth item that hashes to bucket i in the probe sequence starting at bucket i. We get more than j item comparisons when there are items that do not hash to bucket i appearing in the probe sequence before $y_{i,j}$. Thus $E_2 \geq \frac{1}{p} \cdot \sum_{i=0}^{m-1} \sum_{j=1}^{L_i} j$.

Therefore $E_1 \leq E_2$ as wanted.

3. We can show formally that $A_i(k) = A_{m-i-1}(k)$. This suggests that $A_0(k)$=$A_{m-1}(k)$, $A_1(k)$=$A_{m-2}(k)$, etc. If m is odd, we may deduce that (m+1)/2 buckets are probed and when m is even, m/2 buckets are probed.

Why is this a problem? Consider that we may end up filling all (m+1)/2 buckets of a probe sequence with hashed items. If we needed to hash another item to this same sequence, there would appear to be no empty bucket despite only half of the table being used.