Solution for Problem 2 of HW 1.

Proving the correctness of the algorithm. Yes it is correct.

Let's first discuss what this algorithm is not about. It is not about setting i and j to all possible combinations between 1 and n, each time comparing the two length-n substrings of x and y starting from positions i and j, to see if the substrings are equal. That approach would try $\Theta(n^2)$ pairs of starting positions, possibly making $\Theta(n)$ comparisons for each pair, and so would only establish an $O(n^3)$ running time, not the $O(n)$ (linear) time bound claimed.

At the high level, the given algorithm does choose *some* i, j pairs (i and j between 1 and n), and for each pair it does compare the two substrings of x and y starting from those positions. Of course, if those substrings match for n characters, then the algorithm has discovered that α is a circular shift of β. Since that is the only way the algorithm declares that one of the strings is a circular shift of the other, if the algorithm declares that α is a circular shift of β, it actually is one. So the needed thing to prove is the converse: We assume that α is a circular shift of β, and prove that the algorithm will declare this to be so.

The key is to concentrate on the lexicographically-smallest, length-n substring of x (and hence also of y). Call that substring $\alpha*$ and let $i*$ be its leftmost starting position in x, and let $j*$ be its leftmost starting position in y. Clearly, both $i*$ and j^* are in the range 1 to n.

Whenever i is incremented, it is after the algorithm finds that the n-length-substring of y starting at j is lexicographically smaller than the n-length substring of x starting at i. That is then a constructive demonstration that i is not equal to $i*$ (assuming that α is a circular shift of β). Moreover, if i is set to $i + k$ in an iteration, then each n-length substring of y starting at a position between j and $j + k - 1$ is lexicographically smaller than the respective n- length substring starting at a position between i and $i + k - 1$. Hence none of the indices between (the unincremented) i and $i+k-1$ can be $i*$. The same analysis holds when j is set to $j+k$. It follows that if one string is a circular shift of the other, then neither i nor j can be set beyond $i*$ or $j*$ respectively. Hence neither can be set beyond n. But the algorithm only declares that α is not a circular shift of β when one of i or j is set beyond n. Hence, if it terminates, the algorithm will terminate by declaring that α is a circular shift of β. Further, the algorithm must terminate, because in any iteration that does k comparisons, the algorithm either terminates or increments either i or j by k, and neither can go beyond n. This finishes the

proof of correctness.

For the time analysis, again note that in any iteration that does k comparisons, either i or j get incremented by k. The algorithm terminates when both i and j have been set to $n + 1$, or before that, so the algorithm can make at most $2n$ comparisons. Comparisons are the primitive operations, since the total number of operations done in the algorithm is proportional to the number of comparisons. Hence the algorithm runs in $O(n)$ time.