1. **Randomized Algorithms.** A Hamiltonian Cycle is a cycle in an undirected graph that visits each vertex exactly once. Determining whether a graph has a Hamiltonian Cycle is a well-known NP-Complete problem, which means that determining whether a subset of the set of edges in a graph forms a Hamiltonian Cycle is easy, but finding such set might require testing all possible subsets (of which there are $2^{|E|}$). Implement a randomized algorithm to try to find a Hamiltonian cycle in a graph. To do this, you should do the following:

```python
Randomized Hamiltonian(V,E)
    for i in range(maximum trials):
        let Eh be a random subset of E of size V
        if graph (V,Eh) has 1 connected component and the in-degree of every vertex in V is 2
            return Eh # Eh forms a Hamiltonian cycle
    return None  # No Hamiltonian cycle was found
```

2. **Backtracking** Implement a backtracking algorithm to solve the Hamiltonian Cycle problem. Make your implementation as efficient as possible.

3. **Dynamic Programming** Modify the edit distance function provided in class to allow replacements only in the case where the characters being interchanged are both vowels, or both consonants. For example, ‘a’ can be replaced by ‘e’ but not by ‘s’, and ‘t’ can be replaced by ‘w’ but not by ‘u’.

As usual, write a report describing your work.