Programming and Data Structures with Python: Problem Set 2

1. Program an in-place version of the insertion sorting algorithm. Is your implementation stable? Check.

2. Implement the iterative quicksort algorithm where the left index sweeps from left to right and the right index sweeps from right to left (with the invariant that everything to the left of the left index is smaller than the pivot and everything to the right of the right index is bigger than the pivot). Test your implementation on a number of inputs.

3. Modify the above implementation so that it uses a comparison function as a parameter.

4. Let l be a list of strings over the letters $A - Za - z$. Write a Python function `bucket(l,i)` which returns a list, say ls constructed as follows: First transfer the elements of l that do not have position i (i.e. whose length is $< i + 1$) to ls. Then transfer all the elements that have letter A at position i to the list ls. Then transfer those with the letter B at position i and so on till letter z. Here are some examples to illustrate how this function works:

   ```
   bucket(["Abc","dEf","Baec","dA"],0) = ["Abc","Baec","dEF","dA"]
   bucket(["Abc","dEf","Baec","dA"],2) = ["dA", "dEF","Baec","Abc"]
   bucket(["Abc","dEf","Baec","dA"],3) = ["dA", "dEF","Abc", "Baec"]
   bucket(["Abc","dEf","Baec","dA"],7) = ["Abc","dEf","Baec","dA"]
   ```

   [You have to look up the Python documentation to see how to iterate over the list of characters $'A'...'z'$.]

   Write a sorting function `BucketSort(l)` that sorts lists of strings over the alphabet $A-Za-z$ by repeatedly calling the function `bucket`.

   What is the complexity of your function? Why ?

5. Write a iterative python function `sumfree(n)` which returns all sum-free list sublists of $[1, 2, \ldots n]$. That is, $[a_1, a_2, \ldots a_k]$ for some k, with $1 \leq a_1 < a_2 \ldots < a_k \leq n$ such that for any $1 \leq i, j \leq k$, $i \neq j$ we have $a_i + a_j$ is not in the list $[a_1, ...a_k]$.

   ```
   sumfree(3) = [[],[1,2],[1],[2], [3], [2.3], [1,3]]
   ```

   The list $[1, 2, 4]$ is in `sumfree(4)` but $[1, 3, 4]$ is not. The order of the lists in your answer is not important. So you may choose any that is convenient to you. Your program will only be tested on small inputs, say $n \leq 12$.

6. We can define an ordering on the set of permutations of $1..n$, by comparing them lexicographically. In other words, for permutations σ and π ($\sigma \neq \pi$), let i_0 be the first position where they differ. Then $\sigma < \pi$ if $\sigma[i_0] < \pi[i_0]$ and $\sigma > \pi$ otherwise.

   Given a permutation σ, the next permutation is the least permutation among all permutations greater than σ.

Write a Python function `nextPermutation` which takes a list of integers as argument and returns the next permutation of the given permutation, or an empty list if it does not exist. You may assume that the given list is permutation of $1..n$.

```
nextPermutation([1,2,3]) = [1,3,2]
nextPermutation([3,2,1]) = []
nextPermutation([1,2,8,7,6,5,4,3]) = [1,3,2,4,5,6,7,8]
```

What is the complexity of your function? Why ?
