Programming Questions (50 marks):

The North American Student Tracking Association (NASTA) maintains and operates on multiple lists of *n* students. Each student is identified by a **unique** 8-digit code, called StudentIDentificationCode (SIDC); (e.g. # SIDC: 47203235). Some of these student lists are local to villages, towns and remote areas, where *n* counts only to few hundred students, and possibly less. Others are at the urban cities or provincial levels, where *n* counts to tens of thousands or more.

NASTA needs your help to design a clever "student tracking" ADT called **CleverSIDC**. Keys of CleverSIDC entries are long integers of 8 digits, and one can retrieve the keys/values of an CleverSIDC or access a single element by its key. Furthermore, similar to *Sequences*, given a CleverSIDC key, one can access its predecessor or successor (if it exists).

CleverSIDC adapts to their usage and keep the balance between memory and runtime requirements. For instance, if an CleverSIDC contains only a small number of entries (e.g., few hundreds), it might use less memory overhead but slower (sorting) algorithms. On the other hand, if the number of contained entries is large (greater than 1000 or even in the range of tens of thousands of elements), it might have a higher memory requirement but faster (sorting) algorithms. CleverSIDC might be almost constant in size or might grow and/or shrink dynamically. Ideally, operations applicable to a single CleverSIDC entry should be *O(1)* but never worse than *O(n)*. Operations applicable to a complete CleverSIDC should not exceed $O(n^2)$.

You have been asked to **design** and **implement** the CleverSIDC ADT, which automatically adapts to the dynamic content that it operates on. In other words, it accepts the size (total number of students, *n*, identified by their 8 digits SIDC number as a key) as a parameter and uses an appropriate (set of) data structure(s), or other data types, from the one(s) studied in class in order to perform the operations below efficiently[1]. You are NOT allowed however to use any of the built-in data types (that is, you must implement whatever you need, for instance, linked lists, expandable arrays, hash tables, etc. yourself).

The **CleverSIDC** must implement the following methods:

- **SetSIDCThreshold (Size)**: where 100 ≤ Size ≤ ~500,000 is an integer number that defines the size of the list. This size is very important as it will determine what data types or data structures will be used (i.e. a Tree, Hash Table, AVL tree, binary tree, sequence, etc.);
- **generate()**: randomly generates new non-existing key of 8 digits;
- **allKeys(CleverSIDC)**: return all keys in CleverSIDC as a **sorted sequence;**
- **add(CleverSIDC,key,value[2])**: add an entry for the given key and value;
- **remove(CleverSIDC,key)**: remove the entry for the given key;
- **getValues(CleverSIDC,key)**: return the values of the given key;

[1] The lower the memory and runtime requirements of the ADT and its operations, the better will be your marks.

[2] Value here could be any info of the student. You can use a single string composed of Family Name, First Name, and DOB.

- **nextKey(CleverSIDC,key)**: return the key for the successor of key;
- **prevKey(CleverSIDC,key)**: return the key for the predecessor of key;
- **rangeKey(key1, key2)**: returns the number of keys that are within the specified range of the two keys *key1* and *key2*.

1. Write the pseudo code for at least 4 of the above methods.
2. Write the java code that implements all the above methods.
3. Discuss how both the *time* and *space* complexity change for each of the above methods depending on the underlying structure of your CleverSIDC (i.e. whether it is an array, linked list, etc.)?

You have to submit the following deliverables:

a) A detailed report about your design decisions and specification of your CleverSIDC ADT including a rationale and comments about assumptions and semantics.

b) Well-formatted and documented Java source code and the corresponding class files with the implemented algorithms.

c) Demonstrate the functionality of your CleverSIDC by documenting at least 5 different, but representative, data sets. These examples should demonstrate all cases of your CleverSIDC ADT functionality (e.g**., all operations of your ADT for different sizes).** You have to additionally test your implementation with benchmark files that are posted along with the assignment.