1. **Double Hashing Map:** Implement Map interface by using an open addressing hash table. Collisions are resolved by double hashing.
2. **Recursive Hashing Set:** Implement Set interface by using a chaining hash table. The reference of the value is stored in the table if there is no collision. Collisions are resolved by inserting the values (mapped to the same table entry) into another hash table of same type. So, the structure is recursive.
3. **MergeSort with DoubleLinkedList :** Implement merge sort with double linked list. Analyze the running time of your method.
4. **Average Run Time Analysis:** Collect data about the average running-time of each algorithm discussed in class. Create arrays of random integers and then measure the time it takes to sort for each algorithm. Repeat this process, for at least 10 different size of arrays and at least 10 times for each array size. Report the average runtime (draw graphs) for each algorithm. (Hint: System.nanoTime() gives you a nanosecond-precise time)
5. **Worst-case Performance Analysis:** In order to compare worst-case performance of sort algorithms (discussed in class) perform the following experiment: For each algorithm generate input arrays giving worst-case performance. Generate arrays of at least 4 different sizes:100, 1000, 5000, 10000. Report the performance analysis and show that graphics.

*each algorithm: 5 sorting algorithms
Question1:MergeSort with dll
Four algorithms in books codes: Insertion Sort, Merge Sort, Quick Sort, HeapSort.

Q4 average run time analysis and Q5 worst-case performance analysis steps must be applied to all algorithms (5 sorting algorithm)

Note:
- Obey OOP principles
- Use intelliJ IDE on the given VM. VM download link can be found on moodle(in HW1)
- Your submission is studentnumber.zip and include following files:
- o intelliJ project file
    - Q1 folder
    - Q2 folder
    - Q3 folder
    - Q4 folder
    - Q5 folder
- o Report.pdf
- o Javadoc
- The report must be in format "ReportFormat_hw5.doc"