Create RESTful Spring Boot app which will implement below algorithms and return results based on given integer number.  
Use separate endpoint for getting result of each conversion. Pay attention to result's numeral system.  
**Important** **note:**The algorithm code have to be written only using Java API from JDK 11 or above - this requirement is for algorithm code ONLY. The web layer should of course be written using frameworks and libraries.

**Algorithms:**

Given not negative decimal integer number **N** has to be converted to binary system and then:

* return the number in binary system as a string.  
  Example:  
  Input: 55  
  Output: { "result": "110111**"**}
* switch digits (**1** to **0**,  **0** to **1,** e.g. **1000** to **0111**) and return the number in octal system as a string.  
  Example:  
  Input: 55  
  Output: { "result": "10**"**}
* reverse the digits (e.g. **0100** to **0010**) and return the number in hexadecimal system as a string.  
  Example:  
  Input: 55  
  Output: { "result": "3B**"**}
* move all 1s to the left, and all 0s to the right (e.g. **1010** to **1100**) and return the number in decimal system as a string.  
  Example:  
  Input: 55  
  Output: { "result": "62**"**}

Please ignore all **zeros** from the left side before the first **one**in all conversions e.g. decimal 4 is binary 100 and decimal 8 is binary 1000.

**Requirements:**

* the API has to return data in JSON format only
* the whole project should have short documentation with instructions how to run that application and with the list of available endpoints.
* use Gradle (preferably) or Maven as build tool
* solution should be uploaded to GitHub or any other public repository

**Stretch:**

* Save input data and results to database and create sample endpoint which will list all results from database
* Use API documentation framework (e.g. Swagger)
* Dockerise the application and publicise to e.g. docker hub, so it can be run locally using Docker without building it (add instructions how to run it). Docker-compose would be the best for that one.
* Write unit tests