**Lab Tasks**

**Q1.** Generate an array of 500 random integers in the range: [1-10]. Then find the following:

1. Print the number of occurrences for each integer in the original array (1-10).
2. Take user input for k. Find the kth largest integer in the array, and print the number of occurrences in the original array. (If k = 1 this means the largest integer).
3. Take user input for a value to search. Then replace each occurrence with a new value, which will also be provided by the user input.

**Q2.** Generate an array of 10000 random integers and perform merge sort. Compare the performance of merge sort with an elementary bubble sort in terms of **time taken**.

**Q3.** Generate a dynamic 2D array with dimensions specified on runtime. Initialise the array with random integers, ranging from [1-1000]. Take a user input for integer to search and implement a binary search algorithm to efficiently find the integer. Keep in mind that binary search only works consistently for **sorted arrays**, and also that the algorithm will need to be slightly adapted since **the array is 2D**.

**Q4.** Refer back to Q2. Perform the same implementation, with additional functions for quick and radix sort. Compare the performance of these 3 sorting algorithms (merge, quick, radix) on an array of 10000 random integers. **Perform the simulation 100 times**, each time initialising new values in the array. Then compare the results in terms of best, worst, and average times for each algorithm.

**Q5.** Binary & Interpolation search could perform better or worse than each other depending on the uniformity of data distribution. A uniform data distribution is one where all the possible data values have the same frequency.

Generate an array of 10000 random integers, in the range [1-100]. Create a function which measures the **“uniformity”** of the data and returns a float value. **Perform a simulation 100 times**, each time initialising new values in the array. Then call the function to measure uniformity, and call both binary & interpolation search algorithms to find a certain value [1-100] in the array. Compare the results and print them in tabular form, showing the performance (time taken) of binary & interpolation search in comparison to the **uniformity** of the array.

**Note:** Create your function for measuring uniformity based on your own logic. There is no “perfect” answer, many different methods could be implemented according to your understanding.