Department of Computer Science and Engineering,
Anna University, Chennai- 600025
CS23302 – Data Structures and Algorithms
Practical
July – November 2026
Year/Sem/Batch : II/III/ N

Exercise: 13	Sorting and Hashing	16.10.2026

Observation (5 Marks)

1. Trace selection sort on 64, 25, 12, 22, 11 and show the array after every pass. How many swaps does it make? How many swaps would bubble sort make on the same input? Which of the two would you use when writing to memory is expensive, for example on flash storage?
2. Sort 170, 45, 75, 90, 802, 24, 2, 66 using LSD radix sort. Show the array after each pass (units, tens and hundreds digit). Why must the sort used inside each pass be stable? State the time complexity of radix sort for n keys of d digits in base b.
3. (a) Quick sort always picks the first element as the pivot. How many comparisons between the pivot and the other elements are made when the input is the already sorted array 1, 2, 3, 4, 5, 6? Write the recurrence for this case. (b) What is the minimum number of comparisons that any comparison-based algorithm needs in the worst case to sort 5 distinct elements? Justify your answer using a decision tree.
4. Insert the keys 72, 27, 36, 24, 63, 81, 92, 101 into a hash table of size 10 with h(k) = k mod 10 using (a) linear probing, (b) quadratic probing and (c) separate chaining. Draw the final table for each method and write the number of collisions for every key. Which method suffers from primary clustering?
5. Insert the keys 14, 17, 25, 37, 34, 16, 26 into a hash table of size 11 using double hashing with h1(k) = k mod 11 and h2(k) = 7 – (k mod 7). Show the probe sequence for every key that collides and draw the final table. Why must h2(k) never evaluate to 0, and why is the table size usually chosen to be a prime number?
6. Define load factor. When and how is rehashing performed? Explain primary and secondary clustering. Why can a slot not simply be emptied when a key is deleted from an open-addressing table, and how does a "deleted" (tombstone) marker solve this problem?

Execution (15 Marks)

1. Implement a Railway Platform Planner for a busy junction using sorting.

   Define a Train structure that contains:
   - TrainNo (string)
   - Route (string)
   - Arrival (integer, 24-hour time in hhmm format, e.g. 0940)
   - Departure (integer, hhmm)

a. Sort the trains by arrival time using merge sort. If two trains arrive at the same time, the one that departs earlier comes first.

b. Write a function minPlatforms() that sorts the arrival times and the departure times separately and uses two pointers to find the minimum number of platforms needed so that no train waits. A train that arrives at or before the departure time of another train needs a separate platform.

c. Write a function assignPlatforms() that takes the trains in arrival order and gives each train the lowest-numbered free platform (the previous train on that platform must have departed strictly before the new train arrives).

d. Write a main() function that performs the following operations:

(i) Store the 8 trains below (they are deliberately not in order).

Train No	Route	Arrival	Departure
T104	Chennai – Tirupati	1100	1130
T101	Chennai – Madurai	0900	0910
T106	Chennai – Mysuru	1800	2000
T102	Coimbatore – Chennai	0940	1200
T107	Salem – Chennai	1100	1145
T103	Bengaluru – Chennai	0950	1120
T105	Madurai – Chennai	1500	1900
T108	Chennai – Trichy	1130	1215

(ii) Display the schedule sorted by arrival time.

(iii) Display the minimum number of platforms required.

(iv) Display the platform allotted to each train.

(v) Train T103 is delayed by 40 minutes (it now arrives at 1030 and departs at 1200). Update the record, recompute, and display how many platforms are needed now.

2. Implement a FASTag Toll Plaza Registry using a hash table with separate chaining.

Define a Vehicle structure that contains:

- RegNo (string, e.g. TN01AB1234)
- OwnerName (string)
- Balance (integer, in rupees)

a. Use the hash function computed with Horner's rule: start with h = 0 and, for every character c of RegNo, set h = (h × 31 + c) mod TableSize.

b. Implement the following operations:

- insert(Vehicle v) – add a vehicle to the registry
- search(RegNo) – if the vehicle is found, deduct the toll of ₹85 and display the new balance, or display "Insufficient balance" if the balance is less than ₹85
- delete(RegNo) – remove a blacklisted vehicle
- display() – print every bucket with its chain and chain length, and the current load factor

c. Start with a table of size 7. After every insertion, if the load factor exceeds 0.75, rehash all vehicles into a new table whose size is the smallest prime number that is at least twice the old size.

d. Write a main() function that performs the following operations:

(i) Insert the 8 vehicles below and display the table just before and just after the rehash.

Reg No	Owner	Balance (₹)
TN01AB1234	Ravi Kumar	450
KA05MN4321	Sneha Rao	120
TN09CD5678	Arjun Das	60
KL07EF2468	Meera Nair	300
TN22GH1357	Farook Ali	85
AP16JK9999	Kiran Reddy	500
TN10LM2020	Priya S	40
TN38PQ7777	Vignesh M	250

(ii) Search for KA05MN4321, TN22GH1357, TN10LM2020 and an unregistered vehicle TN99ZZ0001, and display the result of each search.

(iii) Delete KL07EF2468 and display the table.

(iv) Report the longest chain before and after rehashing and explain how rehashing affects the search time.

3. Write C programs for the following three coding-round problems. Print the input and the output for every test case.

a. Largest number: arrange the given non-negative integers so that, written side by side, they form the largest possible number. Use qsort() with a custom comparator that places x before y if the string xy is greater than yx.

Input: `3 30 34 5 9`
Expected output: `9534330`
Input: `10 2`
Expected output: `210`
Input: `0 0`
Expected output: `0`

b. Minimum swaps to sort: find the minimum number of swaps needed to sort an array of distinct integers. Sort a copy of the array, use a hash table to map each value to its correct position, and count the cycles.

Input: `4 3 2 1`
Expected output: `2`
Input: `1 5 4 3 2`
Expected output: `2`
Input: `10 19 6 3 5`
Expected output: `2`

c. Longest consecutive sequence: find the length of the longest run of consecutive integers in an unsorted array in O(n) time using a hash set, without sorting. Explain why a sorting-based solution would take O(n log n).

Input:	`100 4 200 1 3 2`
Expected output:	`4  (1, 2, 3, 4)`
Input:	`0 3 7 2 5 8 4 6 0 1`
Expected output:	`9`