**VARUN BHARGAVA – 241010282**

**DATA STRUCTURES TASK-5**

**Task 01: Hashing:**

( https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab4/lab4-1.c )

1. Implement a hash table for storing student roll numbers using the division

method of hashing.

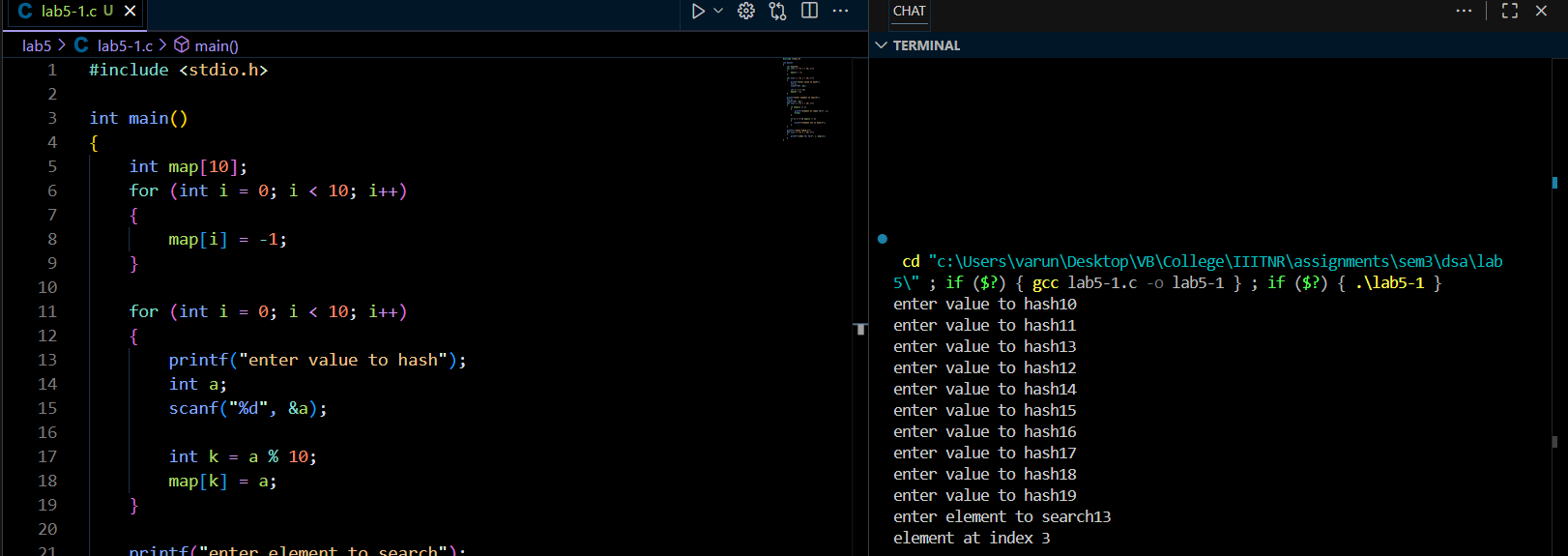
Use hash function: Index = Roll\_no % table\_size

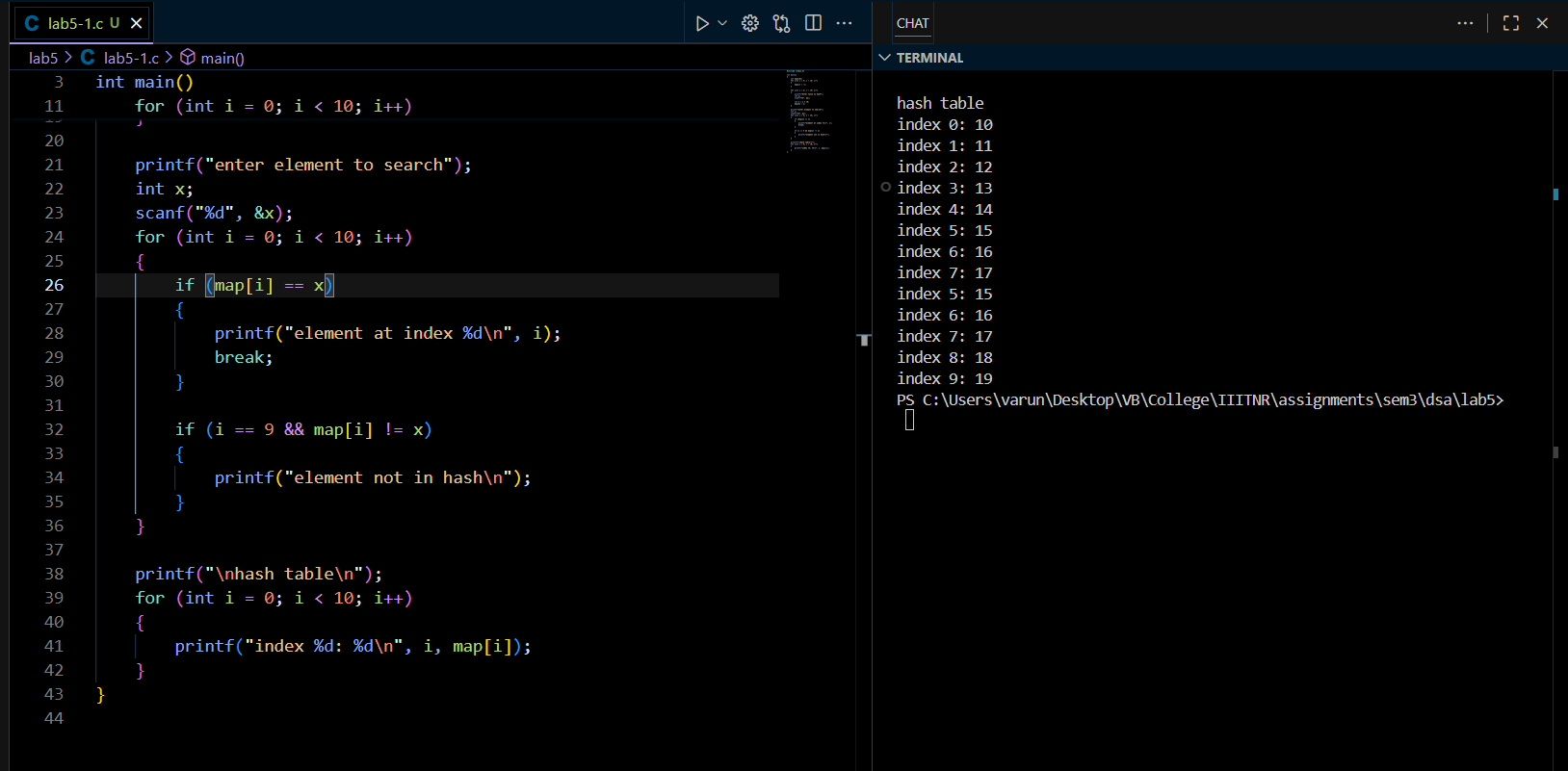
● Insert 10 roll numbers into the hash table (assume no collisions).

Roll no.: 10, 21, 32, 43, 54, 65, 76, 87, 98, 109

● Search for a given roll number and display its index.

● Display/Print the hash table.





**Task 02: Chaining:**

( https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab4/lab4-2.c)

Use hash function : Index = Integer\_keys % 7

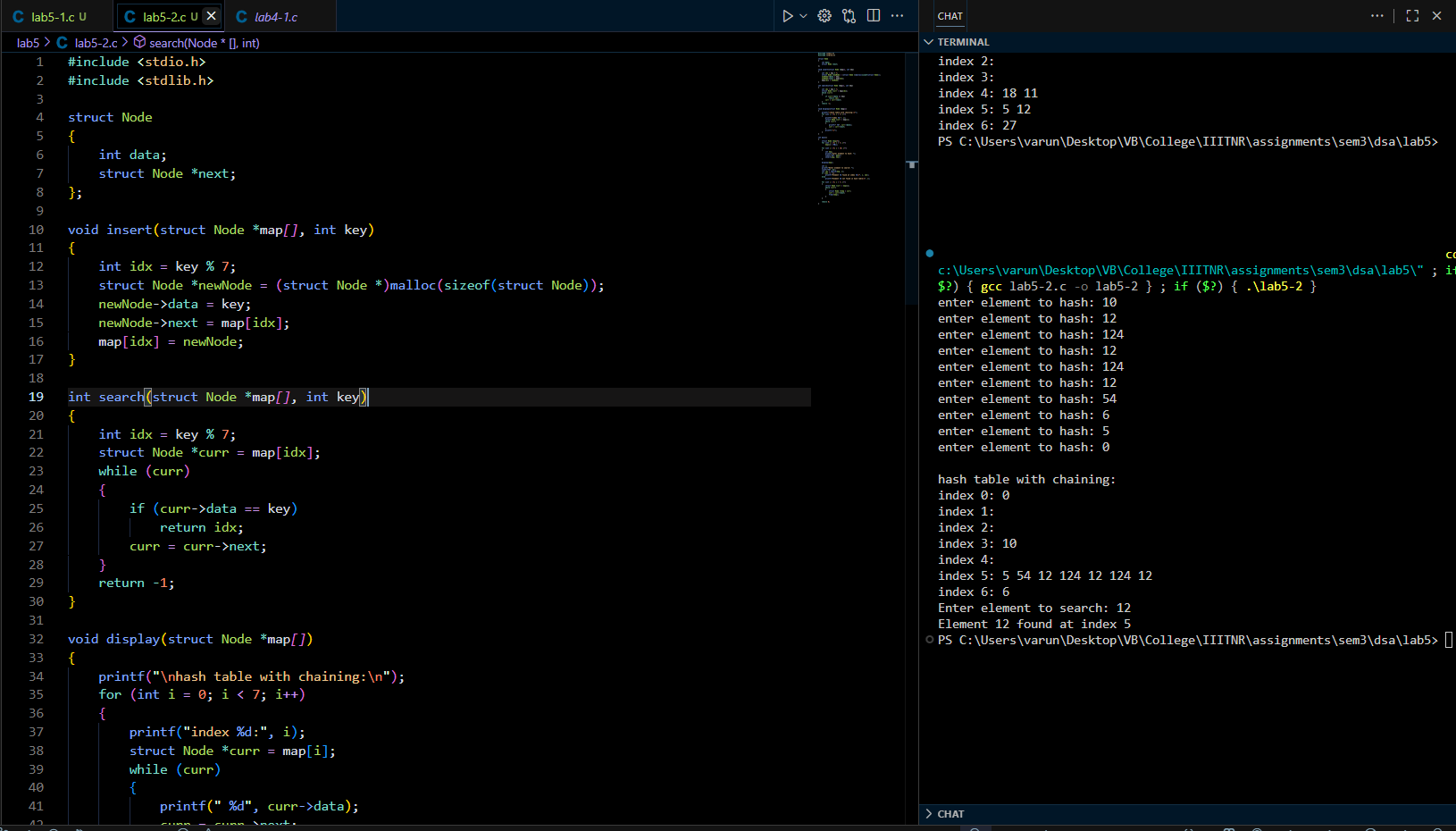
● Use linked lists to handle collisions. If collision occurs insert at the beginning of

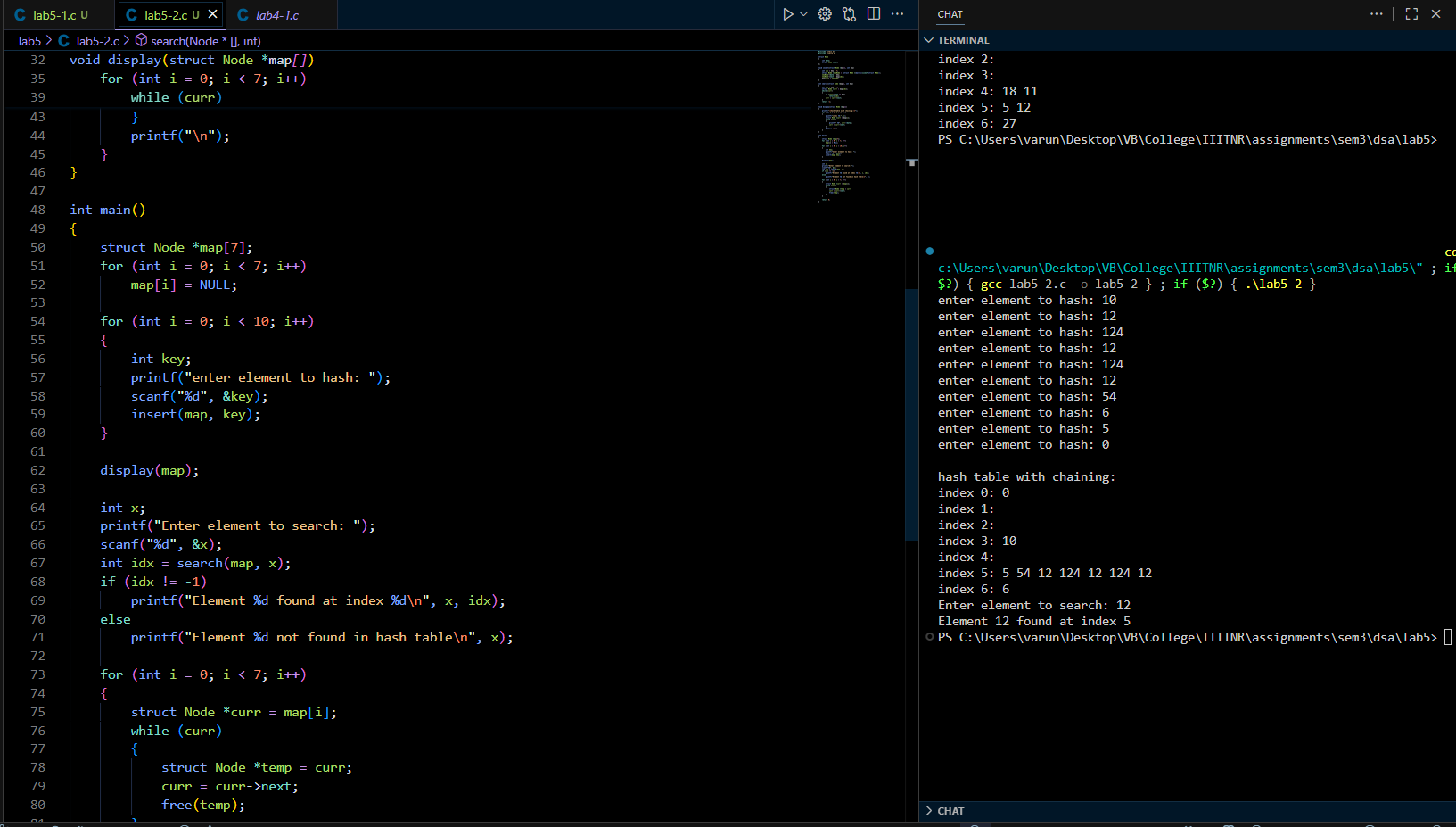
the chain(Linked list).

● Insert 10 integer keys.

Integer keys : 15, 11, 27, 8, 12, 14, 5, 7, 18, 29

● Display/Print the hash table with chains.





**Task 03: Linear Probing:**

( https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab4/lab4-3.c )

Use hash function : Index = Integer\_keys % 10

● Insert 10 integer keys using linear probing.

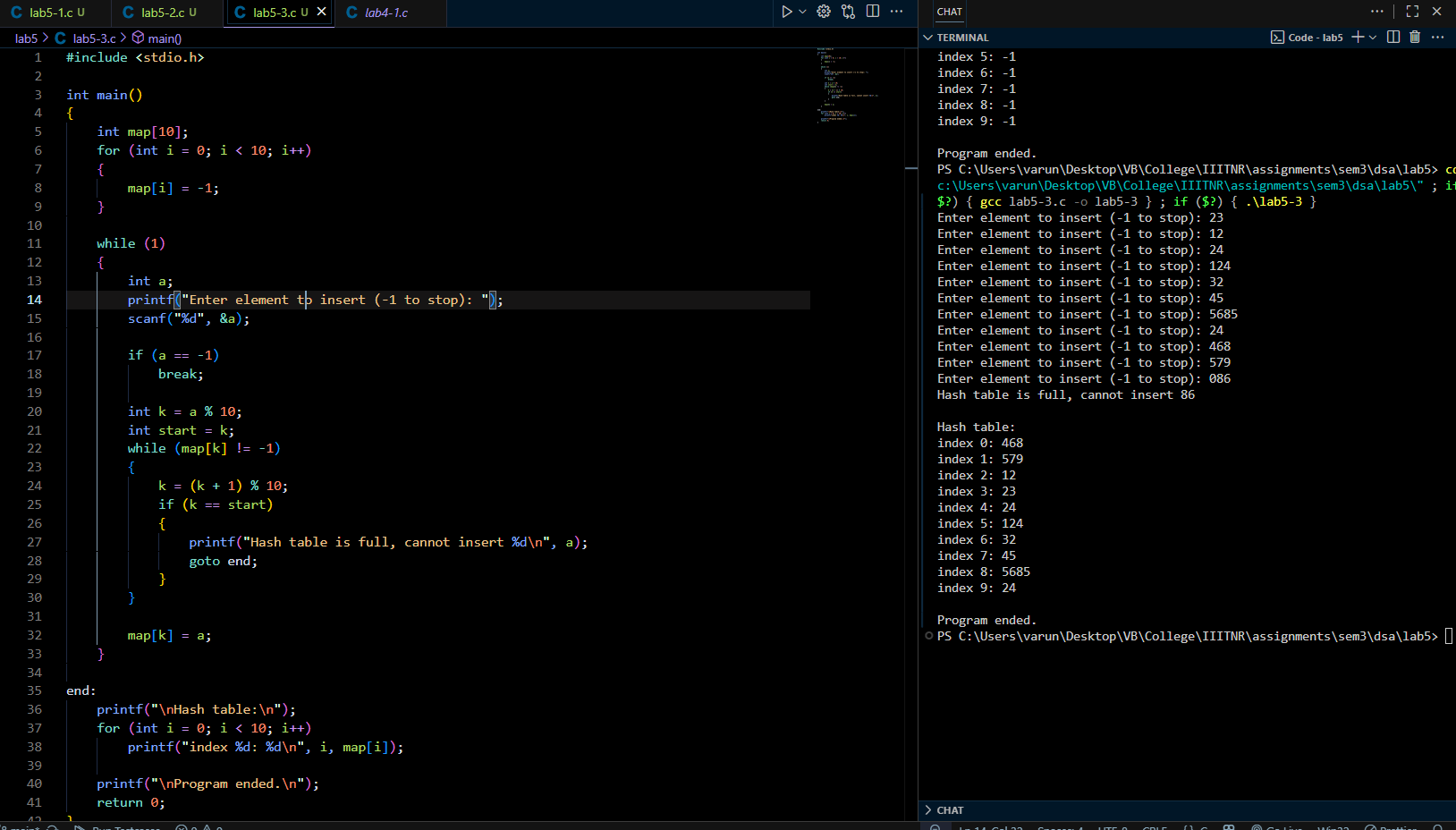
Integer keys : 23, 43, 13, 27, 39, 54, 31, 72, 18, 29

● Show how collisions are resolved by checking the next available slot.

1. Resolve collisions by checking the next available slot in a linear sequence.

2. If a slot is occupied, move to the next slot (index + 1) % 10.

● Print the final hash table.



**Task 04: Quadratic Probing:**

( https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab4/lab4-3.c )

Use hash function : Index = Integer\_keys % 11

● Insert 10 integer keys using quadratic probing.

Integer keys : 19, 27, 36, 10, 64, 29, 20, 55, 39, 75

● Resolve collisions using quadratic steps.

If the index is occupied, try (h(Integer\_keys) + i^2) % 11 increment i = 1, 2, 3, ...

until an empty slot is found (where h is the hash function.)

● Print the final hash table.

