## Aim

To implement hashing using **linear probing** method for collision resolution.

---

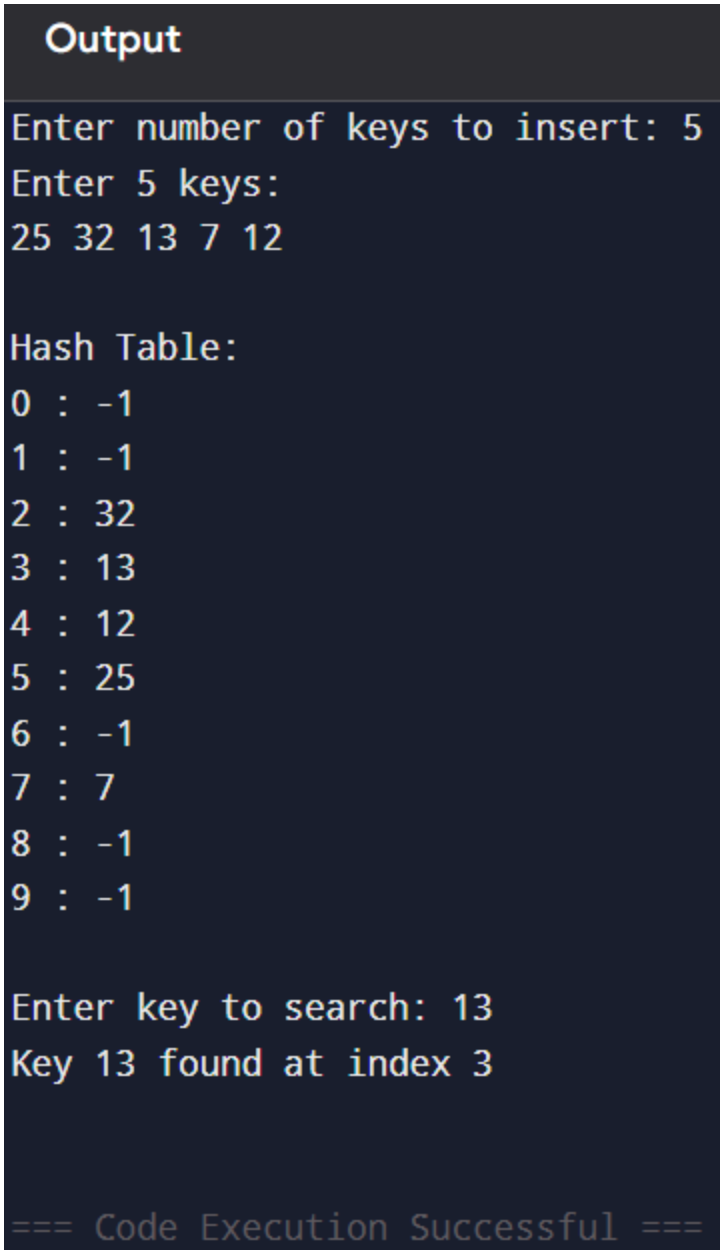
## ⚙️ Algorithm

1. Start with an empty hash table of fixed size.
2. For each key to insert:
    - Compute index = key % table_size.
    - If slot empty → insert.
    - Else → move linearly (index+1, index+2, …) until empty slot found.
3. To search a key:
    - Compute index = key % table_size.
    - Probe linearly until key is found or an empty slot is reached.

# CODE:

```
#include <stdio.h>
#define SIZE 10

int hashTable[SIZE];

void initTable() {
   for (int i = 0; i < SIZE; i++)
      hashTable[i] = -1;
}

void insert(int key) {
   int index = key % SIZE;
   int i = 0;
   while (hashTable[(index + i) % SIZE] != -1)
      i++;
   hashTable[(index + i) % SIZE] = key;
}
```

```
int search(int key) {
    int index = key % SIZE;
    int i = 0;
    while (hashTable[(index + i) % SIZE] != -1) {
        if (hashTable[(index + i) % SIZE] == key)
            return (index + i) % SIZE;
        i++;
        if (i == SIZE) return -1; // table full
    }
    return -1;
}

void display() {
    printf("\nHash Table:\n");
    for (int i = 0; i < SIZE; i++)
        printf("%d : %d\n", i, hashTable[i]);
}

int main() {
    int n, key, choice;
    initTable();

    printf("Enter number of keys to insert: ");
    scanf("%d", &n);
    printf("Enter %d keys:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &key);
        insert(key);
    }

    display();

    printf("\nEnter key to search: ");
    scanf("%d", &key);
    int pos = search(key);
    if (pos != -1)
        printf("Key %d found at index %d\n", key, pos);
    else
        printf("Key %d not found\n", key);

    return 0;
}
```




**RESULT:**

**The program successfully executed and displayed the hash table using linear probing.**