Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

**Department of Electronics & Telecommunication Engineering**
**Data Structures & Algorithms Lab (DJ19ECSBL1)**

## Data Structures & Algorithms Project report

| Sr. No. | SAP ID | Name of student |
|---|---|---|
| 1 | 60002200155 | Nihal Shaikh |
| 2 | 60002200156 | Abhay Bhosle |
| 3 | 60002200157 | Shweta Joshi |

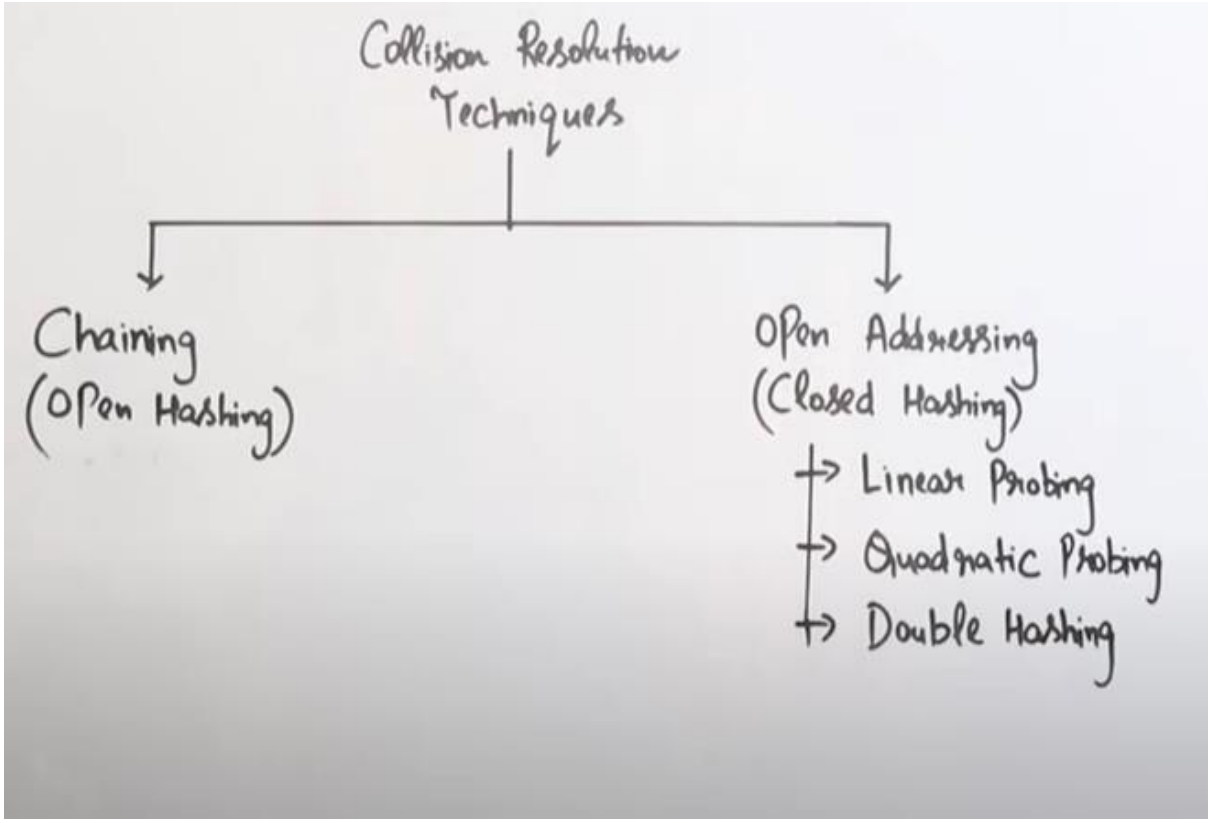
Aim: **Write a code in C to search an element in hash table using linear hashing technique.**

Programming Language: **C++**

**Theory:**

**Hashing:**

Hashing is a technique or process of mapping keys, and values into the hash table by using a hash function. It is done for faster access to elements. The efficiency of mapping depends on the efficiency of the hash function used.




**Linear Hashing:**

linear probing, the hash table is searched sequentially that starts from the original location of the hash. If in case the location that we get is already occupied, then we check for the next location.

**Implementation:**

Let **hash(x)** be the slot index computed using a hash function and **S** be the table size

If slot hash(x) % S is full, then we try (hash(x) + 1) % S

If (hash(x) + 1) % S is also full, then we try (hash(x) + 2) % S

If (hash(x) + 2) % S is also full, then we try (hash(x) + 3) % S

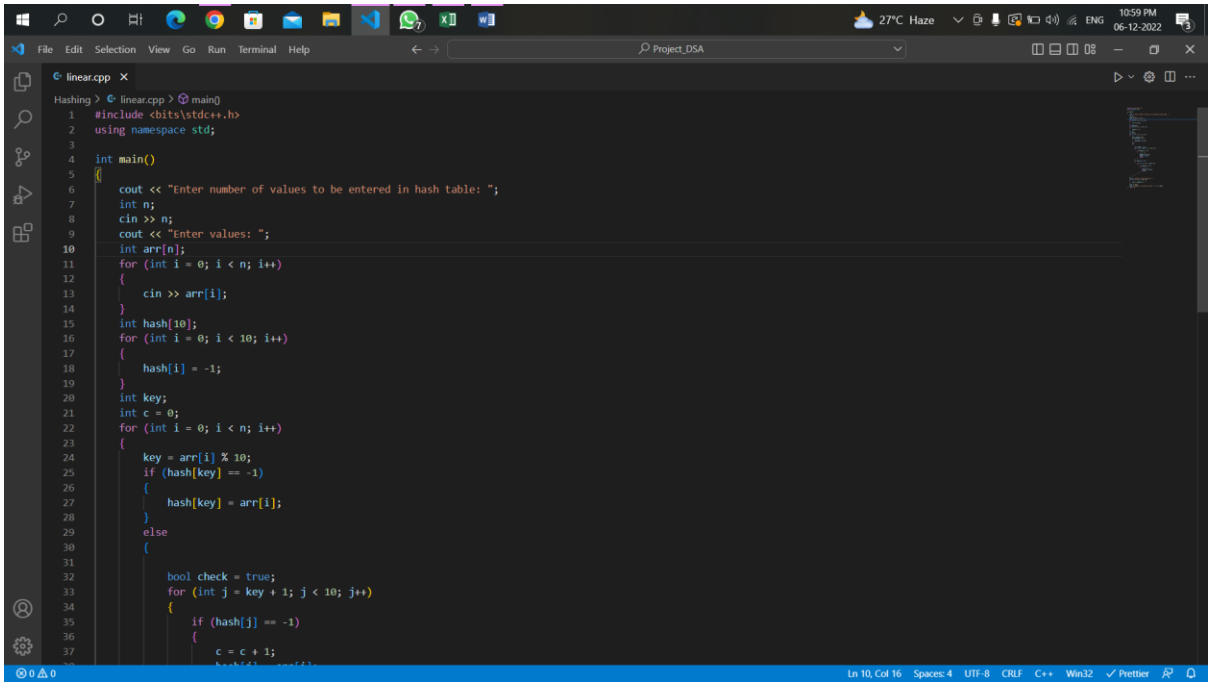
**Code:**

```
#include <bits\stdc++.h>
using namespace std;

int main()
{
   cout << "Enter number of values to be entered in hash table: ";
   int n;
   cin >> n;
   cout << "Enter values: ";
   int arr[n];
   for (int i = 0; i < n; i++)
   {
      cin >> arr[i];
   }
   int hash[10];
   for (int i = 0; i < 10; i++)
   {
      hash[i] = -1;
   }
   int key;
   int c = 0;
   for (int i = 0; i < n; i++)
   {
      key = arr[i] % 10;
      if (hash[key] == -1)
      {
         hash[key] = arr[i];
      }
      else
      {

         bool check = true;
         for (int j = key + 1; j < 10; j++)
         {
            if (hash[j] == -1)
            {
               c = c + 1;
               hash[j] = arr[i];
               check = false;
               break;
            }
         }
```

```
        if (check == true)
        {
           for (int j = 0; j < key; j++)
           {
              if (hash[j] == -1)
              {
                 c = c + 1;
                 hash[j] = arr[i];
                 break;
              }
           }
        }
     }
  }
  cout << "Values in hash table are: ";
  for (int i = 0; i < 10; i++)
  {
     cout << hash[i] << " ";
  }
  cout << endl;
  cout << "Number of collision occurred: " << c << endl;
  return 0;
}
```

## Result:

Implementation of the code:



```cpp
#include <bits\stdc++.h>
using namespace std;

int main()
{
    cout << "Enter number of values to be entered in hash table: ";
    int n;
    cin >> n;
    cout << "Enter values: ";
    int arr[n];
    for (int i = 0; i < n; i++)
    {
        cin >> arr[i];
    }
    int hash[10];
    for (int i = 0; i < 10; i++)
    {
        hash[i] = -1;
    }
    int key;
    int c = 0;
    for (int i = 0; i < n; i++)
    {
        key = arr[i] % 10;
        if (hash[key] == -1)
        {
            hash[key] = arr[i];
        }
        else
        {

            bool check = true;
            for (int j = key + 1; j < 10; j++)
            {
                if (hash[j] == -1)
                {
                    c = c + 1;
```

## Zero collisions:

```
PS G:\programming\Project_DSA\Hashing> cd "g:\programming\Project_DSA\Hashing\" ; if ($?) { g++ linear.cpp -o linear } ; if ($?) { .\linear }
Enter number of values to be entered in hash table: 5
Enter values: 1 2 3 4 5
Values in hash table are: -1 1 2 3 4 5 -1 -1 -1 -1
Number of collision occured: 0
PS G:\programming\Project_DSA\Hashing>
```

## Maximum collisions:

```
PS G:\programming\Project_DSA> cd "g:\programming\Project_DSA\Hashing\" ; if ($?) { g++ linear.cpp -o linear } ; if ($?) { .\linear }
Enter number of values to be entered in hash table: 5
Enter values: 21 31 41 51 61
Values in hash table are: -1 21 31 41 51 61 -1 -1 -1 -1
Number of collision occured: 4
PS G:\programming\Project_DSA\Hashing>
```

## Disadvantage of linear hashing is its size:

No extra number can have a position if the hash table is full.

```
PS G:\programming\Project_DSA\Hashing> cd "g:\programming\Project_DSA\Hashing\" ; if ($?) { g++ linear.cpp -o linear } ; if ($?) { .\linear }
Enter number of values to be entered in hash table: 11
Enter values: 0 1 2 3 4 5 6 7 8 9 10
Values in hash table are: 0 1 2 3 4 5 6 7 8 9
Number of collision occured: 0
PS G:\programming\Project_DSA\Hashing>
```

**Conclusion:**

1) **Primary Clustering:** Primary clustering is one of the issues with linear probing. Many successive items form clusters, making it difficult to locate a free slot or to search for an element.
2) **Secondary Clustering:** Secondary clustering is less severe, and two records can only share a collision chain (also known as a probe sequence) if they start out in the same location.
3) **K%10** hash function was implemented for **linear hashing.**
4) Linear hashing helps to locate values according to their keys very efficiently.
5) We can access element with the help of keys without traversing like array. Whose time complexity becomes **O(1) to access values**.
6) **Space** is one of the **constraints** in **linear hashing.**