1.Harshing

Insert 10,12,23,42,53,62,74,85,96,105,116 in harshing table

```
#include <stdio.h>

#define TABLE_SIZE 10

// Function to calculate the hash value
int hashFunction(int key) {
    return key % TABLE_SIZE;
}

// Function to insert a value into the hash table using linear probing
void insert(int hashTable[], int key) {
    int index = hashFunction(key);

    // Probe linearly until an empty spot is found
    while (hashTable[index] != -1) {
        index = (index + 1) % TABLE_SIZE;
    }

    hashTable[index] = key;
}

// Function to display the hash table
void display(int hashTable[]) {
    printf("Final Hash Table:\n");
```

```
    for (int i = 0; i < TABLE_SIZE; i++) {
        if (hashTable[i] == -1) {
            printf("Index %d: [Empty]\n", i);
        } else {
            printf("Index %d: %d\n", i, hashTable[i]);
        }
    }
}

int main() {
    int hashTable[TABLE_SIZE];

    // Initialize all positions in hash table to -1 (indicating empty slots)
    for (int i = 0; i < TABLE_SIZE; i++) {
        hashTable[i] = -1;
    }

    // Predefined elements to insert
    int elements[] = {10, 12, 23, 42, 53, 62, 74, 85, 96, 105, 116};
    int n = sizeof(elements) / sizeof(elements[0]);

    // Insert elements into the hash table
    for (int i = 0; i < n; i++) {
        insert(hashTable, elements[i]);
    }
```

```
    // Display the hash table
    display(hashTable);

    return 0;
}
```

Output

```
Final Hash Table:
Index 0: 10
Index 1: 105
Index 2: 12
Index 3: 23
Index 4: 74
Index 5: 85
Index 6: 96
Index 7: 116
Index 8: 42
Index 9: 53
```

2. Linear probing insert 79,17,47,58,69,32,97

```
#include <stdio.h>

#define TABLE_SIZE 10

// Function to calculate the hash value
Int hashFunction(int key) {
    Return key % TABLE_SIZE;
```

```
}

// Function to insert a value into the hash table using linear probing
Void insert(int hashTable[], int key) {
   Int index = hashFunction(key);

   // Probe linearly until an empty spot is found
   While (hashTable[index] != -1) {
      Index = (index + 1) % TABLE_SIZE;
   }

   hashTable[index] = key;
}

// Function to display the hash table
Void display(int hashTable[]) {
   Printf(“Final Hash Table:\n”);
   For (int I = 0; I < TABLE_SIZE; i++) {
      If (hashTable[i] == -1) {
         Printf(“Index %d: [Empty]\n”, i);
      } else {
         Printf(“Index %d: %d\n”, I, hashTable[i]);
      }
   }
}
```

```
Int main() {
   Int hashTable[TABLE_SIZE];

   // Initialize all positions in hash table to -1 (indicating empty slots)
   For (int I = 0; I < TABLE_SIZE; i++) {
      hashTable[i] = -1;
   }

   Int n;
   Printf("Enter the number of elements to insert: ");
   Scanf("%d", &n);

   Int elements[n];
   Printf("Enter the elements to insert:\n");
   For (int I = 0; I < n; i++) {
      Printf("Element %d: ", I + 1);
      Scanf("%d", &elements[i]);
   }

   // Insert elements into the hash table
   For (int I = 0; I < n; i++) {
      Insert(hashTable, elements[i]);
   }

   // Display the hash table
   Display(hashTable);
```

```
    Return 0;
}
```

Input:

Enter the number of elements to insert: 7

Enter the elements to insert:

Element 1: 79

Element 2: 17

Element 3: 47

Element 4: 58

Element 5: 69

Element 6: 32

Element 7: 97

Output:

Final Hash Table:

Index 0: 58

Index 1: 69

Index 2: 32

Index 3: 97

Index 4: [Empty]

Index 5: [Empty]

Index 6: [Empty]

Index 7: 17

Index 8: 47

Index 9: 79