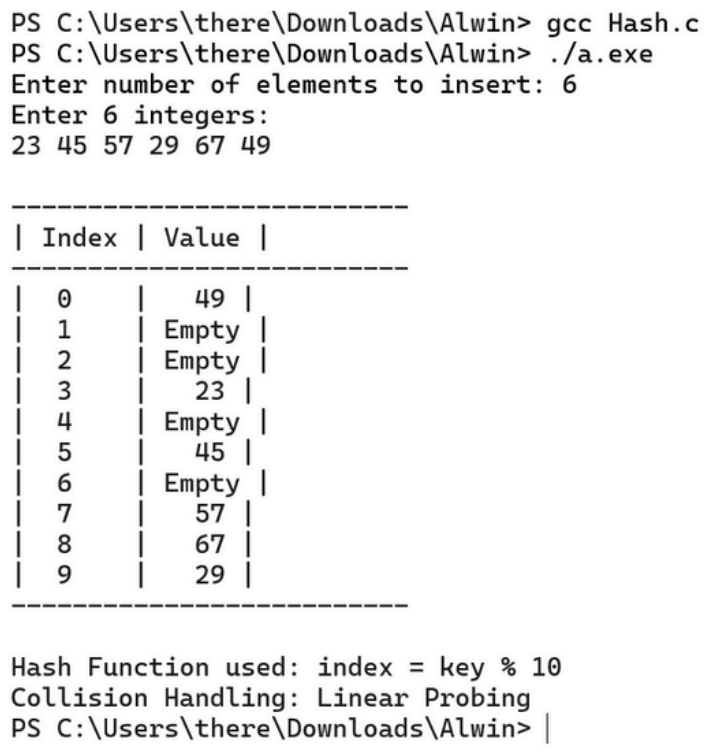
EXPERIMENT-14

Program :

```
#include <stdio.h>
// Bubble Sort
void bubbleSort(int a[], int n) {
    for (int i = 0; i < n-1; i++)
        for (int j = 0; j < n-i-1; j++)
            if (a[j] > a[j+1]) {
                int t = a[j];
                a[j] = a[j+1];
                a[j+1] = t;
            }
}
// Insertion Sort
void insertionSort(int a[], int n) {
    for (int i = 1; i < n; i++) {
        int key = a[i], j = i - 1;
        while (j >= 0 && a[j] > key)
            a[j+1] = a[j--];
        a[j+1] = key;
    }
}
// Quick Sort
int partition(int a[], int low, int high) {
    int pivot = a[high], i = low - 1;
    for (int j = low; j < high; j++)
        if (a[j] < pivot) {
            i++;
            int t = a[i]; a[i] = a[j]; a[j] = t;
        }
    int t = a[i+1]; a[i+1] = a[high]; a[high] = t;
    return i+1;
}
void quickSort(int a[], int low, int high) {
    if (low < high) {
```

```c
    if (low < high) {
        int pi = partition(a, low, high);
        quickSort(a, low, pi-1);
        quickSort(a, pi+1, high);
    }
}
// Merge Sort
void merge(int a[], int l, int m, int r) {
    int n1 = m-l+1, n2 = r-m;
    int L[n1], R[n2];
    for (int i = 0; i < n1; i++) L[i] = a[l+i];
    for (int j = 0; j < n2; j++) R[j] = a[m+1+j];

    int i=0, j=0, k=l;
    while (i<n1 && j<n2)
        a[k++] = (L[i] <= R[j]) ? L[i++] : R[j++];
    while (i<n1) a[k++] = L[i++];
    while (j<n2) a[k++] = R[j++];
}
void mergeSort(int a[], int l, int r) {
    if (l < r) {
        int m = (l + r)/2;
        mergeSort(a, l, m);
        mergeSort(a, m+1, r);
        merge(a, l, m, r);
    }
}
// Print array
void printArray(int a[], int n) {
    for (int i = 0; i < n; i++)
        printf("%d ", a[i]);
    printf("\n");
}
// Main function
int main() {
    int a[] = {5, 2, 9, 1, 5, 6};
    int n = sizeof(a)/sizeof(a[0]);
    int b1[6], b2[6], b3[6], b4[6];
    for (int i = 0; i < n; i++)
        b1[i] = b2[i] = b3[i] = b4[i] = a[i];
    bubbleSort(b1,n);
    insertionSort(b2,n);
    quickSort(b3,0,n-1);
    mergeSort(b4,0,n-1);
    printf("Original Array: ");
    printArray(a,n);
    printf("Bubble Sort:    "); printArray(b1,n);
    printf("Insertion Sort: "); printArray(b2,n);
    printf("Quick Sort:     "); printArray(b3,n);
    printf("Merge Sort:     "); printArray(b4,n);

    return 0;
}#include <stdio.h>
#define SIZE 10

// Simple hash function
int hashFunction(int key) {
    return key % SIZE;
}

int main() {
    int hashTable[SIZE];
    int n, key, index;

    // Initialize hash table
    for (int i = 0; i < SIZE; i++)
        hashTable[i] = -1;

    // Input number of elements
    printf("Enter number of elements to insert: ");
    scanf("%d", &n);
    printf("Enter %d integers:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &key);
        index = hashFunction(key);
        // Linear probing for collision handling
        while (hashTable[index] ≠ -1) {
            index = (index + 1) % SIZE;
        }

        hashTable[index] = key;
    }

    // Print the hash table
    printf("\n--------------------------\n");
    printf("| Index | Value |\n");
    printf("--------------------------\n");
    for (int i = 0; i < SIZE; i++) {
        if (hashTable[i] ≠ -1)
            printf("| %2d    | %4d |\n", i, hashTable[i]);
        else
            printf("| %2d    | Empty |\n", i);
    }
    printf("--------------------------\n");
    printf("\nHash Function used: index = key %% 10\n");
    printf("Collision Handling: Linear Probing\n");
    return 0;
}
```

Output :



```
PS C:\Users\there\Downloads\Alwin> gcc Hash.c
PS C:\Users\there\Downloads\Alwin> ./a.exe
Enter number of elements to insert: 6
Enter 6 integers:
23 45 57 29 67 49

--------------------------
| Index | Value |
--------------------------
|  0    |   49 |
|  1    | Empty |
|  2    | Empty |
|  3    |   23 |
|  4    | Empty |
|  5    |   45 |
|  6    | Empty |
|  7    |   57 |
|  8    |   67 |
|  9    |   29 |
--------------------------

Hash Function used: index = key % 10
Collision Handling: Linear Probing
PS C:\Users\there\Downloads\Alwin> |
```