```c
#include <stdio.h>
#define MAX 20

int hashTable[MAX];
int m;

/* Initialize hash table */
void initialize() {
    int i;
    for (i = 0; i < m; i++)
        hashTable[i] = -1;
}

/* Insert key using linear probing */
void insert(int key) {
    int index = key % m;

    if (hashTable[index] == -1) {
        hashTable[index] = key;
    } else {
        int i = (index + 1) % m;
        while (i != index) {
            if (hashTable[i] == -1) {
                hashTable[i] = key;
                return;
            }
            i = (i + 1) % m;
        }
        printf("Hash Table is Full. Cannot insert %d\n", key);
    }
}

/* Display hash table */
void display() {
    int i;
    printf("\nHash Table:\n");
    for (i = 0; i < m; i++) {
        if (hashTable[i] != -1)
            printf("Address %d : %d\n", i, hashTable[i]);
        else
            printf("Address %d : Empty\n", i);
    }
}
```

```
int main() {
    int n, key, i;

    printf("Enter size of hash table (m): ");
    scanf("%d", &m);

    initialize();

    printf("Enter number of employee records: ");
    scanf("%d", &n);

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

    display();
    return 0;
}
```

Input
Enter size of hash table (m): 10
Enter number of employee records: 5
Enter 5 employee keys:
1234
5678
9012
1244
5688

Output
Hash Table:
Address 0 : 9012
Address 1 : Empty
Address 2 : 1244
Address 3 : 1234
Address 4 : Empty
Address 5 : Empty
Address 6 : Empty
Address 7 : 5678
Address 8 : 5688

Address 9 : Empty