| S.No: 13 | Exp. Name: ***Quadratic Probing Technique*** | Date:2025-10-18 |
|---|---|---|





## Aim:

Smith is enhancing the record management system to use quadratic probing for handling collisions while hashing with a fixed-size hash table. The records are identified by four-digit keys, and Smith wants to efficiently insert these records into the hash table. Your task is to help Smith to write a C program which creates a hash table, inserts records using quadratic probing, and displays the contents.

**Input Format:**

- First line of input contains two space separated integers $N$, $m$ representing number if keys and size of the hash table
- Second line of input contains $N$ space separated four-digit integers representing the values.

**Output Format:**

- If the hash table is empty after all operations, print:

```
Hash Table is empty
```

- Otherwise, for each index $i$ of the hash table (from 0 to $m$-1), print:

```
T[i] -> <value>
```

where value is the value stored at index $i$.

- If the hash table becomes full and a key cannot be inserted, print:

```
Hash table is full. Cannot insert records from key {X}
```

where $X$ is the 1-based position of the key that couldn't be inserted.

**Note:**

- Refer to the sample test cases for better understanding of input and output formats.

## Source Code:

HashTable1.c

```
// Type your content here...
#include <stdio.h>
#include <stdlib.h>
void insert(int ht[], int m, int key, int count){
   int index = key % m;
   int orgi = index;
   while(ht[index]!=-1){
      index = (index+1)%m;
      if(index == orgi){
         printf("Hash table is full. Cannot insert records from key %d\n", count);
         return;
      }
   }
   ht[index] = key;
}
int main(){
   int n,m;
```

```
    int count = 0;
    scanf("%d %d", &n, &m);
    if(n==0){
       printf("Hash Table is empty\n");
       return 0;
    }
    int keys[n];
    for(int i = 0; i<n; i++){
       scanf("%d", &keys[i]);
    }
    int ht[m];
    for(int i = 0; i<m; i++)
       ht[i]=-1;
    for(int i = 0; i<n; i++){
       count++;
       insert(ht, m, keys[i], count);
       if(count > m)
          break;
    }
    for(int i = 0; i<m; i++){
       printf("T[%d] -> %d\n", i, ht[i]);
    }
    printf("\n");

    return 0;
}
```

## Execution Results - All test cases have succeeded!

| Test Case - 1 |
|---|
| User Output |
| 5 5 |
| 1596 3574 9512 7532 4268 |
| T[0] -> 4268 |
| T[1] -> 1596 |
| T[2] -> 9512 |
| T[3] -> 7532 |
| T[4] -> 3574 |

| Test Case - 2 |
|---|
| User Output |
| 6 4 |
| 4587 6589 4521 6523 5824 6548 |
| Hash table is full. Cannot insert records from key 5 |
| T[0] -> 6523 |
| T[1] -> 6589 |
| T[2] -> 4521 |
| T[3] -> 4587 |

| Test Case - 3 |
|---|
| User Output |

| 0 5 |
| --- |
| Hash Table is empty |