DATA STRUCTURES

Division	CS(AIML) -A
Batch	2
GR-no	12311493
Roll no	54
Name	Atharva Kangralkar

Assignment 15:

Implement following collision handling techniques for Hash table. A. Linear probing. B. Quadratic Probing. C. Double Hashing using Mod as a Hash Function.

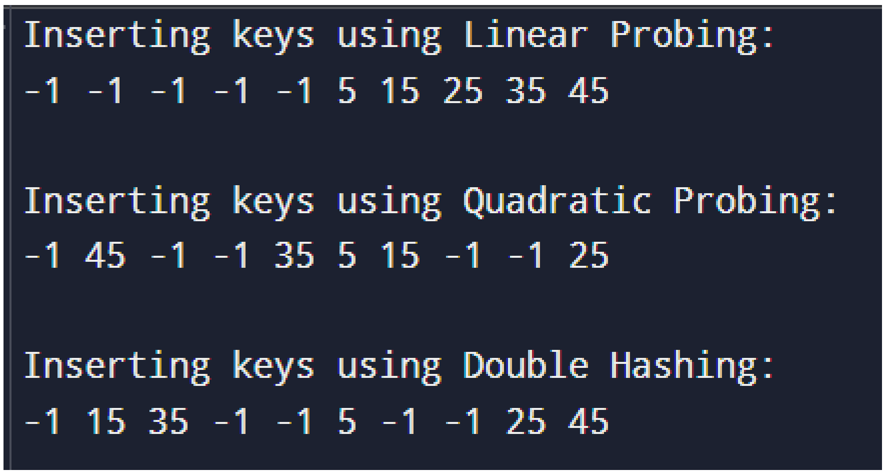
Code:-

```
#include <stdio.h>

#include <stdlib.h>


#define TABLE_SIZE 10


struct HashTable {

   int *table;

};


void initHashTable(struct HashTable *hashTable) {

   hashTable->table = (int *)malloc(sizeof(int) * TABLE_SIZE);

   for (int i = 0; i < TABLE_SIZE; i++) {

      hashTable->table[i] = -1;

   }

}


int hash(int key) {

   return key % TABLE_SIZE;

}


int linearProbing(struct HashTable *hashTable, int key) {

   int index = hash(key);

   int i = 0;

   while (hashTable->table[(index + i) % TABLE_SIZE] != -1) {

      i++;

   }

   return (index + i) % TABLE_SIZE;
```

```
}

int quadraticProbing(struct HashTable *hashTable, int key) {
   int index = hash(key);
   int i = 0;
   while (hashTable->table[(index + i * i) % TABLE_SIZE] != -1) {
      i++;
   }
   return (index + i * i) % TABLE_SIZE;
}

int doubleHashing(struct HashTable *hashTable, int key) {
   int index = hash(key);
   int stepSize = 7 - (key % 7);
   int i = 0;
   while (hashTable->table[(index + i * stepSize) % TABLE_SIZE] != -1) {
      i++;
   }
   return (index + i * stepSize) % TABLE_SIZE;
}

void insertLinear(struct HashTable *hashTable, int key) {
   int index = linearProbing(hashTable, key);
   hashTable->table[index] = key;
}

void insertQuadratic(struct HashTable *hashTable, int key) {
   int index = quadraticProbing(hashTable, key);
```

```
    hashTable->table[index] = key;
}

void insertDoubleHashing(struct HashTable *hashTable, int key) {
    int index = doubleHashing(hashTable, key);
    hashTable->table[index] = key;
}

void displayHashTable(struct HashTable *hashTable) {
    for (int i = 0; i < TABLE_SIZE; i++) {
        printf("%d ", hashTable->table[i]);
    }
    printf("\n");
}

int main() {
    struct HashTable hashTable;
    initHashTable(&hashTable);

    printf("Inserting keys using Linear Probing:\n");
    insertLinear(&hashTable, 5);
    insertLinear(&hashTable, 15);
    insertLinear(&hashTable, 25);
    insertLinear(&hashTable, 35);
    insertLinear(&hashTable, 45);
    displayHashTable(&hashTable);

    initHashTable(&hashTable);
    printf("\nInserting keys using Quadratic Probing:\n");
    insertQuadratic(&hashTable, 5);
    insertQuadratic(&hashTable, 15);
    insertQuadratic(&hashTable, 25);
    insertQuadratic(&hashTable, 35);
    insertQuadratic(&hashTable, 45);
    displayHashTable(&hashTable);

    initHashTable(&hashTable);

    printf("\nInserting keys using Double Hashing:\n");
    insertDoubleHashing(&hashTable, 5);
    insertDoubleHashing(&hashTable, 15);
    insertDoubleHashing(&hashTable, 25);
    insertDoubleHashing(&hashTable, 35);
    insertDoubleHashing(&hashTable, 45);
    displayHashTable(&hashTable);

    return 0;
}
```

Output:-



```
Inserting keys using Linear Probing:
-1 -1 -1 -1 -1 5 15 25 35 45

Inserting keys using Quadratic Probing:
-1 45 -1 -1 35 5 15 -1 -1 25

Inserting keys using Double Hashing:
-1 15 35 -1 -1 5 -1 -1 25 45
```