```
/*******************************ASSIGNMENT 3********************************
Consider the telephone book database of N clients. Make use of a hash table
implementation to quickly look up a client’s telephone number. Make use of
linear probing, double hashing and quadratic collision handling techniques.


Name- Tanishq Jaywant Pawar
Roll no-41
batch-S2
********************************************************************************/






#include <stdio.h>
#define SIZE 10


struct HashTable {
  int index;
  long mobile;
};


// Global hash table
struct HashTable h[SIZE];



// Initialize hash table
void initialize() {
  for (int i = 0; i < SIZE; i++) {
      h[i].index = i;
      h[i].mobile = -1;
  }
}


//Display HAsh Table


void display() {
  printf("\nHash Table:\n");
  for (int i = 0; i < SIZE; i++) {
      printf("Index %d: %ld\n", h[i].index, h[i].mobile);
  }
}


//Insert telephone number by using Linear Probing
int linearProbing(int pos) {
  int i = pos;
  do {
    if (h[i].mobile == -1)
      return i;
    i = (i + 1) % SIZE;
  } while (i != pos);
  return -1; // table is full
}


//Insert telephone number by using Quadratic probing


int quadraticProbing(long key) {
  int a;
  for (int j = 0; j < SIZE; j++) {
    a = (key + (j * j)) % SIZE;
    if (h[a].mobile == -1)
      return a;
  }
  return -1; // table is full
}


// Insert telephone number by using Double Hashing
int hash2(long key) {
  return 7 - (key % 7);        // Make sure SIZE and 7 are relatively prime
}


int doubleHashing(long key) {
  int hash1 = key % SIZE;
  int hash2Val = hash2(key);
  int i = 0, newIndex;


  while (i < SIZE) {
    newIndex = (hash1 + i * hash2Val) % SIZE;
    if (h[newIndex].mobile == -1)
      return newIndex;
    i++;
  }
  return -1;              // table is full
}


void insert(int method) {
 long key;
  int pos, newPos;


  printf("Enter mobile number to insert: ");
   scanf("%ld", &key);
```

```
    pos = key % SIZE;


    if (h[pos].mobile == -1) {
        h[pos].mobile = key;
    } else {
        if (method == 1) {
            newPos = linearProbing(pos);
        } else if (method == 2) {
            newPos = quadraticProbing(key);
        } else {
            newPos = doubleHashing(key);
        }


        if (newPos != -1) {
            h[newPos].mobile = key;
        } else {
            printf("Hash table is full, cannot insert!\n");
        }
    }
}


void search() {
    long key;
    int pos, found = 0;


    printf("Enter mobile number to search: ");
    scanf("%ld", &key);


    pos = key % SIZE;


    if (h[pos].mobile == key) {
        printf("Mobile number %ld found at index %d\n", key, pos);
        return;
    }


    for (int i = 0; i < SIZE; i++) {
        if (h[i].mobile == key) {
            printf("Mobile number %ld found at index %d\n", key, i);
            found = 1;
            break;
        }
    }


    if (!found) {
        printf("Mobile number %ld not found in the hash table\n", key);
```

```
  }
}


int main() {
  int ch, method;


  initialize();


  do {
      printf("\nMenu:\n");
      printf("1. Insert\n");
      printf("2. Display\n");
      printf("3. Search\n");
      printf("4. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &ch);


      switch (ch) {
          case 1:
              printf("Collision Methods:\n");
              printf("1. Linear Probing\n2. Quadratic Probing\n3. Double Hashing\n");
              printf("Choose collision handling method:");
              scanf("%d", &method);
              insert(method);
              break;
          case 2:
              display();
              break;
          case 3:
              search();
              break;
          case 4:
              break;
          default:


              printf("Invalid choice.\n");
      }
  } while (ch != 4);


  return 0;
}
/***********************************OUTPUT****************************************
student@student-OptiPlex-3000:~$ ./ass3


Menu:
1. Insert
```

```
2. Display
3. Search
4. Exit
Enter your choice: 1
Collision Methods:
1. Linear Probing
2. Quadratic Probing
3. Double Hashing
Choose collision handling method:1
Enter mobile number to insert: 7412589632


Menu:
1. Insert
2. Display
3. Search
4. Exit
Enter your choice: 1
Collision Methods:
1. Linear Probing
2. Quadratic Probing
3. Double Hashing
Choose collision handling method:1
Enter mobile number to insert: 7412588741


Menu:
1. Insert
2. Display
3. Search
4. Exit
Enter your choice: 1
Collision Methods:
1. Linear Probing
2. Quadratic Probing
3. Double Hashing
Choose collision handling method:1
Enter mobile number to insert: 8569857412


Menu:
1. Insert
2. Display
3. Search
4. Exit
Enter your choice: 1
Collision Methods:
1. Linear Probing
2. Quadratic Probing
3. Double Hashing
Choose collision handling method:2
Enter mobile number to insert: 8741257411
```

```
Menu:
1. Insert
2. Display
3. Search
4. Exit
Enter your choice: 2


Hash Table:
Index 0: -1
Index 1: 7412588741
Index 2: 7412589632
Index 3: 8569857412
Index 4: -1
Index 5: 8741257411
Index 6: -1
Index 7: -1
Index 8: -1
Index 9: -1


Menu:
1. Insert
2. Display
3. Search
4. Exit
Enter your choice: 1
Collision Methods:
1. Linear Probing
2. Quadratic Probing
3. Double Hashing
Choose collision handling method:2
Enter mobile number to insert: 9874568745


Menu:
1. Insert
2. Display
3. Search
4. Exit
Enter your choice: 1
Collision Methods:
1. Linear Probing
2. Quadratic Probing
3. Double Hashing
Choose collision handling method:2
Enter mobile number to insert: 9632587415


Menu:
1. Insert
2. Display
3. Search
```

```
4. Exit
Enter your choice: 1
Collision Methods:
1. Linear Probing
2. Quadratic Probing
3. Double Hashing
Choose collision handling method:3
Enter mobile number to insert: 7854788996


Menu:
1. Insert
2. Display
3. Search
4. Exit
Enter your choice: 1
Collision Methods:
1. Linear Probing
2. Quadratic Probing
3. Double Hashing
Choose collision handling method:3
Enter mobile number to insert: 7413588741


Menu:
1. Insert
2. Display
3. Search
4. Exit
Enter your choice: 2


Hash Table:
Index 0: -1
Index 1: 7412588741
Index 2: 7412589632
Index 3: 8569857412
Index 4: 7413588741
Index 5: 8741257411
Index 6: 9874568745
Index 7: -1
Index 8: 7854788996
Index 9: 9632587415


Menu:
1. Insert
2. Display
3. Search
4. Exit
Enter your choice: 1
Collision Methods:
1. Linear Probing
```

```
2. Quadratic Probing
3. Double Hashing
Choose collision handling method:3
Enter mobile number to insert: 7412558896


Menu:
1. Insert
2. Display
3. Search
4. Exit
Enter your choice: 1
Collision Methods:
1. Linear Probing
2. Quadratic Probing
3. Double Hashing
Choose collision handling method:3
Enter mobile number to insert: 9987414785


Menu:
1. Insert
2. Display
3. Search
4. Exit
Enter your choice: 2


Hash Table:
Index 0: 7412558896
Index 1: 7412588741
Index 2: 7412589632
Index 3: 8569857412
Index 4: 7413588741
Index 5: 8741257411
Index 6: 9874568745
Index 7: 9987414785
Index 8: 7854788996
Index 9: 9632587415


Menu:
1. Insert
2. Display
3. Search
4. Exit
Enter your choice: 1
Collision Methods:
1. Linear Probing
2. Quadratic Probing
3. Double Hashing
Choose collision handling method:1
Enter mobile number to insert: 745888741
```

```
Hash table is full, cannot insert!


Menu:
1. Insert
2. Display
3. Search
4. Exit
Enter your choice: 3
Enter mobile number to search: 7412589632
Mobile number 7412589632 found at index 2


Menu:
1. Insert
2. Display
3. Search
4. Exit
Enter your choice: 2


Hash Table:
Index 0: 7412558896
Index 1: 7412588741
Index 2: 7412589632
Index 3: 8569857412
Index 4: 7413588741
Index 5: 8741257411
Index 6: 9874568745
Index 7: 9987414785
Index 8: 7854788996
Index 9: 9632587415


Menu:
1. Insert
2. Display
3. Search
4. Exit
Enter your choice: 4

**********************************************************************************/
```