Linear probing

Program

```
#include <stdio.h>
#define TABLE_SIZE 10

int hashTable[TABLE_SIZE];

void initHashTable() {
    for (int i = 0; i < TABLE_SIZE; i++)
        hashTable[i] = -1;
}

int h(int x, int i) {
    return (x % TABLE_SIZE + i) % TABLE_SIZE;
}

void insert(int x) {
    int i = 0, index;
    while (i < TABLE_SIZE) {
        index = h(x, i);
        if (hashTable[index] == -1) {
            hashTable[index] = x;
            printf("Inserted %d at index %d\n", x, index);
            return;
        }
        i++;
    }
    printf("Hash table is full. Could not insert %d\n", x);
}

void search(int x) {
    int i = 0, index;
    while (i < TABLE_SIZE) {
        index = h(x, i);
        if (hashTable[index] == x) {
            printf("Element %d found at index %d\n", x, index);
            return;
        }
        if (hashTable[index] == -1) break;
        i++;
    }
    printf("Element %d not found\n", x);
}

void display() {
    printf("\nHash Table:\n");
```

```
    for (int i = 0; i < TABLE_SIZE; i++)
        printf("%d => %d\n", i, hashTable[i]);
}

int main() {
    int choice, key;
    initHashTable();

    do {
        printf("\n--- LINEAR PROBING MENU ---\n");
        printf("1. Insert\n2. Search\n3. Display\n4. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
        case 1: printf("Enter element: "); scanf("%d", &key); insert(key); break;
        case 2: printf("Enter element: "); scanf("%d", &key); search(key); break;
        case 3: display(); break;
        case 4: printf("Exiting...\n"); break;
        default: printf("Invalid choice\n");
        }
    } while (choice != 4);

    return 0;
}
```

Output

```
--- LINEAR PROBING MENU ---
1. Insert
2. Search
3. Display
4. Exit
Enter choice: 1
Enter element: 4
Inserted 4 at index 4

--- LINEAR PROBING MENU ---
1. Insert
2. Search
3. Display
4. Exit
Enter choice: 1
Enter element: 6
Inserted 6 at index 6

--- LINEAR PROBING MENU ---
1. Insert
2. Search
3. Display
4. Exit
```

```
Enter choice: 2
Enter element: 4
Element 4 found at index 4

--- LINEAR PROBING MENU ---
1. Insert
2. Search
3. Display
4. Exit
Enter choice: 3

Hash Table:
0 => -1
1 => -1
2 => -1
3 => -1
4 => 4
5 => -1
6 => 6
7 => -1
8 => -1
9 => -1
```

Quadratic Probing

Program

```
#include <stdio.h>
#define TABLE_SIZE 10

int hashTable[TABLE_SIZE];

void initHashTable() {
    for (int i = 0; i < TABLE_SIZE; i++)
        hashTable[i] = -1;
}

int h(int x, int i) {
    return (x % TABLE_SIZE + i * i) % TABLE_SIZE;
}

void insert(int x) {
    int i = 0, index;
    while (i < TABLE_SIZE) {
        index = h(x, i);
        if (hashTable[index] == -1) {
            hashTable[index] = x;
            printf("Inserted %d at index %d\n", x, index);
            return;
        }
        i++;
    }
    printf("Hash table is full. Could not insert %d\n", x);
}

void search(int x) {
    int i = 0, index;
    while (i < TABLE_SIZE) {
        index = h(x, i);
        if (hashTable[index] == x) {
            printf("Element %d found at index %d\n", x, index);
            return;
        }
        if (hashTable[index] == -1) break;
        i++;
    }
    printf("Element %d not found\n", x);
}

void display() {
    printf("\nHash Table:\n");
    for (int i = 0; i < TABLE_SIZE; i++)
```

```
        printf("%d => %d\n", i, hashTable[i]);
}

int main() {
    int choice, key;
    initHashTable();

    do {
        printf("\n--- QUADRATIC PROBING MENU ---\n");
        printf("1. Insert\n2. Search\n3. Display\n4. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
        case 1: printf("Enter element: "); scanf("%d", &key); insert(key); break;
        case 2: printf("Enter element: "); scanf("%d", &key); search(key); break;
        case 3: display(); break;
        case 4: printf("Exiting...\n"); break;
        default: printf("Invalid choice\n");
        }
    } while (choice != 4);

    return 0;
}
```

Output

```
--- QUADRATIC PROBING MENU ---
1. Insert
2. Search
3. Display
4. Exit
Enter choice: 1
Enter element: 6
Inserted 6 at index 6

--- QUADRATIC PROBING MENU ---
1. Insert
2. Search
3. Display
4. Exit
Enter choice: 1
Enter element: 8
Inserted 8 at index 8

--- QUADRATIC PROBING MENU ---
1. Insert
2. Search
3. Display
4. Exit
```

```
Enter choice: 2
Enter element: 8
Element 8 found at index 8

--- QUADRATIC PROBING MENU ---
1. Insert
2. Search
3. Display
4. Exit
Enter choice: 3

Hash Table:
0 => -1
1 => -1
2 => -1
3 => -1
4 => -1
5 => -1
6 => 6
7 => -1
8 => 8
9 => -1
```

Double Hashing

Program

```
#include <stdio.h>
#define TABLE_SIZE 10

int hashTable[TABLE_SIZE];

void initHashTable() {
    for (int i = 0; i < TABLE_SIZE; i++)
        hashTable[i] = -1;
}

int h(int x, int i) {
    int h1 = x % TABLE_SIZE;
    int h2 = x % (TABLE_SIZE - 1);
    if (h2 == 0) h2 = 1;
    return (h1 + i * h2) % TABLE_SIZE;
}

void insert(int x) {
    int i = 0, index;
    while (i < TABLE_SIZE) {
        index = h(x, i);
        if (hashTable[index] == -1) {
            hashTable[index] = x;
            printf("Inserted %d at index %d\n", x, index);
            return;
        }
        i++;
    }
    printf("Hash table is full. Could not insert %d\n", x);
}

void search(int x) {
    int i = 0, index;
    while (i < TABLE_SIZE) {
        index = h(x, i);
        if (hashTable[index] == x) {
            printf("Element %d found at index %d\n", x, index);
            return;
        }
        if (hashTable[index] == -1) break;
        i++;
    }
    printf("Element %d not found\n", x);
}
```

```
void display() {
    printf("\nHash Table:\n");
    for (int i = 0; i < TABLE_SIZE; i++)
        printf("%d => %d\n", i, hashTable[i]);
}

int main() {
    int choice, key;
    initHashTable();

    do {
        printf("\n--- DOUBLE HASHING MENU ---\n");
        printf("1. Insert\n2. Search\n3. Display\n4. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
        case 1: printf("Enter element: "); scanf("%d", &key); insert(key); break;
        case 2: printf("Enter element: "); scanf("%d", &key); search(key); break;
        case 3: display(); break;
        case 4: printf("Exiting...\n"); break;
        default: printf("Invalid choice\n");
        }
    } while (choice != 4);

    return 0;
}
```

Output

```
--- DOUBLE HASHING MENU ---
1. Insert
2. Search
3. Display
4. Exit
Enter choice: 1
Enter element: 1
Inserted 1 at index 1

--- DOUBLE HASHING MENU ---
1. Insert
2. Search
3. Display
4. Exit
Enter choice: 1
Enter element: 2
Inserted 2 at index 2

--- DOUBLE HASHING MENU ---
1. Insert
2. Search
3. Display
4. Exit
```

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

Hash Table:
0 => -1
1 => 1
2 => 2
3 => -1
4 => -1
5 => -1
6 => -1
7 => -1
8 => -1
9 => -1
```