CSA0317-DATA STRUCTURES

Program 15

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10
int hashTable[SIZE];
void initializeHashTable() {
   for (int i = 0; i < SIZE; i++) {
      hashTable[i] = -1;
   }
}
int hashFunction(int key) {
   return key % SIZE;
}
void insert(int key) {
   int index = hashFunction(key);
   // Linear probing
   while (hashTable[index] != -1) {
      index = (index + 1) % SIZE;
   }

   hashTable[index] = key;
   printf("Inserted %d at index %d\n", key, index);
}
int search(int key) {
```

```
    int index = hashFunction(key);
    int originalIndex = index;
    // Linear probing search
    while (hashTable[index] != -1) {
        if (hashTable[index] == key) {
            return index;
        }
        index = (index + 1) % SIZE;

        // If we've checked all positions
        if (index == originalIndex) {
            break;
        }
    }
    return -1; // Not found
}

void display() {
    printf("Hash Table:\n");
    for (int i = 0; i < SIZE; i++) {
        if (hashTable[i] != -1) {
            printf("Index %d: %d\n", i, hashTable[i]);
        } else {
            printf("Index %d: Empty\n", i);
        }
    }
```

```
}
int main() {
    initializeHashTable();
    int choice, key, result;
    while (1) {
        printf("\nHashing using Linear Probing\n");
        printf("1. Insert\n");
        printf("2. Search\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter key to insert: ");
                scanf("%d", &key);
                insert(key);
                break;
            case 2:
                printf("Enter key to search: ");
                scanf("%d", &key);
                result = search(key);
                if (result != -1) {
                    printf("Key %d found at index %d\n", key, result);
                } else {
```

```
                printf("Key %d not found\n", key);
            }
            break;
        case 3:
            display();
            break;
        case 4:
            exit(0);
        default:
            printf("Invalid choice!\n");
      }
   }
   return 0;
}
```

Output:

Output Clear

```
Hashing using Linear Probing
1. Insert
2. Search
3. Display
4. Exit
Enter your choice: 1
Enter key to insert: 4
Inserted 4 at index 4

Hashing using Linear Probing
1. Insert
2. Search
3. Display
4. Exit
Enter your choice: 1
Enter key to insert: 6
Inserted 6 at index 6

Hashing using Linear Probing
1. Insert
2. Search
3. Display
4. Exit
Enter your choice: 2
Enter key to search: 6
Key 6 found at index 6

Hashing using Linear Probing
1. Insert
2. Search
3. Display
4. Exit
Enter your choice: 3
Hash Table:
Index 0: Empty
Index 1: Empty
Index 2: Empty
Index 3: Empty
Index 4: 4
Index 5: Empty
Index 6: 6
Index 7: Empty
Index 8: Empty
Index 9: Empty

Hashing using Linear Probing
1. Insert
2. Search
3. Display
4. Exit
Enter your choice: 4


=== Code Execution Successful ===
```