NAME: Venkateswar L
BRANCH: Computer Science and Engineering SEC: F
ROLL NO.: 230701376

PROGRAM: HASHING METHODS

Write a C program to create a hash table and perform collision resolution using the following techniques.

1. Open addressing
2. Closed Addressing
3. Rehashing

OPEN ADDRESSING ALGORITHM

```
#include <stdio.h>

#include <stdlib.h>

struct set {

    int key;

    int data;

};

struct set *array;

int capacity = 10;

int size = 0;

int hashFunction(int key) {

    return (key % capacity);

}

int checkPrime(int n) {

    int i;

    if (n == 1 || n == 0) {
```

```
        return 0;
    }
    for (i = 2; i < n / 2; i++) {
        if (n % i == 0) {
            return 0;
        }
    }
    return 1;
}
void insert(int key, int data) {
    int index = hashFunction(key);
    while (array[index].key != -1) {
        index = (index + 1) % capacity;
    }
    array[index].key = key;
    array[index].data = data;
    size++;
}
int search(int key) {
    int index = hashFunction(key);
    int originalIndex = index;
    while (array[index].key != -1) {
```

NAME: Venkateswar L
BRANCH: Computer Science and Engineering SEC: F
ROLL NO.: 230701376

```
        if (array[index].key == key) {
            return array[index].data;
        }
        index = (index + 1) % capacity;
        if (index == originalIndex) {
            break; // Element not found
        }
    }
    return -1; // Element not found
}
int main() {
    array = (struct set *)malloc(capacity * sizeof(struct set));
    for (int i = 0; i < capacity; i++) {
        array[i].key = -1; // Mark all slots as empty
    }
    insert(42, 100);

    printf("Value for key 42: %d\n", search(42)); // Should print 100

    free(array);
    return 0;
}
```

NAME: Venkateswar L
BRANCH: Computer Science and Engineering SEC: F
ROLL NO.: 230701376

CLOSED ADDRESSING ALGORITHM

```
#include <stdio.h>
#include <stdlib.h>
struct set {
    int key;
    int data;
};
struct set *array;
int capacity = 10;
int size = 0;
int hashFunction(int key) {
    return (key % capacity);
}
void insert(int key, int data) {
    int index = hashFunction(key);
    struct set *newElement = (struct set *)malloc(sizeof(struct set));
    newElement->key = key;
    newElement->data = data;
    newElement->next = NULL;
    if (array[index] == NULL) {
        array[index] = newElement;
```

```
    } else {
        newElement->next = array[index];
        array[index] = newElement;
    }
    size++;
}
int search(int key) {
    int index = hashFunction(key);
    struct set *current = array[index];
    while (current != NULL) {
        if (current->key == key) {
            return current->data;
        }
        current = current->next;
    }
    return -1; // Element not found
}
int main() {
    array = (struct set *)malloc(capacity * sizeof(struct set));
    for (int i = 0; i < capacity; i++) {
        array[i] = NULL; // Initialize each bucket as empty
    }
```

NAME: Venkateswar L
BRANCH: Computer Science and Engineering SEC: F
ROLL NO.: 230701376

```
    insert(42, 100);
     printf("Value for key 42: %d\n", search(42)); // Should print 100
    return 0;
}
```

REHASHING METHOD ALGORITHM

```
#include <stdio.h>
#include <stdlib.h>
#define TABLE_SIZE 11
struct KeyValue {
    int key;
    int value;
};
int hash(int key) {
    return key % TABLE_SIZE;
}
struct KeyValue* createHashTable() {
    struct KeyValue* table = (struct KeyValue*)malloc(TABLE_SIZE *
sizeof(struct KeyValue));
    for (int i = 0; i < TABLE_SIZE; ++i) {
        table[i].key = -1; // Initialize keys to -1 (indicating empty)
```

```
        table[i].value = 0;
    }
    return table;
}
void insert(struct KeyValue* table, int key, int value) {
    int index = hash(key);
    while (table[index].key != -1) {
        index = (index + 1) % TABLE_SIZE;
    }
    table[index].key = key;
    table[index].value = value;
}
int search(struct KeyValue* table, int key) {
    int index = hash(key);
    while (table[index].key != -1) {
        if (table[index].key == key) {
            return table[index].value;
        }
        index = (index + 1) % TABLE_SIZE;
    }
    return -1; // Key not found
}
```

NAME: Venkateswar L
BRANCH: Computer Science and Engineering SEC: F
ROLL NO.: 230701376

```
int main() {
    struct KeyValue* hashTable = createHashTable();
    insert(hashTable, 42, 100);
    printf("Value for key 42: %d\n", search(hashTable, 42));
    free(hashTable);
   return 0;
}
```