#include <stdio.h>

#include <stdlib.h>

#define TABLE\_SIZE 10

struct Node {

int data;

struct Node\* next;

};

typedef struct Node Node;

Node\* hashTable[TABLE\_SIZE];

void initializeTable() {

for (int i = 0; i < TABLE\_SIZE; i++) {

hashTable[i] = NULL;

}

}

int hashFunction(int key) {

return key % TABLE\_SIZE;

}

void insert(int key) {

int hashIndex = hashFunction(key);

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->data = key;

newNode->next = NULL;

if (hashTable[hashIndex] == NULL) {

hashTable[hashIndex] = newNode;

} else {

Node\* temp = hashTable[hashIndex];

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

}

}

int search(int key) {

int hashIndex = hashFunction(key);

Node\* temp = hashTable[hashIndex];

while (temp != NULL) {

if (temp->data == key) {

return 1;

}

temp = temp->next;

}

return 0;

}

void delete(int key) {

int hashIndex = hashFunction(key);

Node\* temp = hashTable[hashIndex];

Node\* prev = NULL;

while (temp != NULL && temp->data != key) {

prev = temp;

temp = temp->next;

}

if (temp == NULL) {

return;

}

if (prev == NULL) {

hashTable[hashIndex] = temp->next;

} else {

prev->next = temp->next;

}

free(temp);

}

void display() {

for (int i = 0; i < TABLE\_SIZE; i++) {

Node\* temp = hashTable[i];

while (temp != NULL) {

printf("%d -> ", temp->data);

temp = temp->next;

}

printf("NULL\n");

}

}

int main() {

int choice, key;

initializeTable();

while (1) {

printf("\nHashing with Collision\n1.Insertion\n2.Search\n3.Delete\n4.Display\n5.Exit\n");

printf("Enter choice:");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter key to insert:");

scanf("%d", &key);

insert(key);

break;

case 2:

printf("Enter to search:");

scanf("%d", &key);

printf(search(key) ? "Found\n" : "Not Found\n");

break;

case 3:

printf("Enter to delete:");

scanf("%d", &key);

delete(key);

break;

case 4:

display();

break;

case 5:

return 0;

}

}

}

Hashing with Collision

1.Insertion

2.Search

3.Delete

4.Display

5.Exit

Enter choice:1

Enter key to insert:1

Hashing with Collision

1.Insertion

2.Search

3.Delete

4.Display

5.Exit

Enter choice:1

Enter key to insert:2

Hashing with Collision

1.Insertion

2.Search

3.Delete

4.Display

5.Exit

Enter choice:1

Enter key to insert:3

Hashing with Collision

1.Insertion

2.Search

3.Delete

4.Display

5.Exit

Enter choice:1

Enter key to insert:4

Hashing with Collision

1.Insertion

2.Search

3.Delete

4.Display

5.Exit

Enter choice:1

Enter key to insert:5

Hashing with Collision

1.Insertion

2.Search

3.Delete

4.Display

5.Exit

Enter choice:2

Enter to search:1

Found

Hashing with Collision

1.Insertion

2.Search

3.Delete

4.Display

5.Exit

Enter choice:3

Enter to delete:5

Hashing with Collision

1.Insertion

2.Search

3.Delete

4.Display

5.Exit

Enter choice:4

NULL

1 -> NULL

2 -> NULL

3 -> NULL

4 -> NULL

NULL

NULL

NULL

NULL

NULL

Hashing with Collision

1.Insertion

2.Search

3.Delete

4.Display

5.Exit

Enter choice:5