1.Implement a hash table of size 10 and 7 using the division method

Code :

#include <iostream>

#include <vector>

#include <list>

using namespace std;

class HashTable {

private:

int size;

vector<list<int>> table;

int hashFunction(int key) {

return key % size;

}

public:

// Correct constructor (no return type)

HashTable(int s) {

size = s;

table.resize(size);

}

void insert(int key) {

int index = hashFunction(key);

table[index].push\_back(key);

}

void display() {

for (int i = 0; i < size; i++) {

cout << i << "-->";

for (int key : table[i]) {

cout << key << " ";

}

cout << endl;

}

}

};

int main() {

int keys[6] = {23, 44, 12, 39, 33, 56};

int n = sizeof(keys) / sizeof(keys[0]);

cout << "Hash table of size 7 : \n";

HashTable ht2(7);

for (int i = 0; i < n; i++) ht2.insert(keys[i]);

ht2.display();

cout<<endl;

cout << "Hash table of size 10 : \n";

HashTable ht1(10);

for (int i = 0; i < n; i++) ht1.insert(keys[i]);

ht1.display();

cout<<"Mehuli Sarkar, av.sc.u4cse24150";

return 0;

}

Output :

A screenshot of a computer code

AI-generated content may be incorrect.

2. From the hash table created in Q1 search for keys {12, 99, 44}.Display if all keys are found or not in its position.

Code :

#include <iostream>

#include <vector>

#include <list>

using namespace std;

class HashTable {

private:

int size;

vector<list<int>> table;

int hashFunction(int key) {

return key % size;

}

public:

HashTable(int s) {

size = s;

table.resize(size);

}

void insert(int key) {

int index = hashFunction(key);

table[index].push\_back(key);

}

bool search(int key, int &index) {

index = hashFunction(key);

// Search for key in the list at table[index]

for (int k : table[index]) {

if (k == key) {

return true;

}

}

return false;

}

void display() {

for (int i = 0; i < size; i++) {

cout << i << "-->";

for (int key : table[i]) {

cout << key << " ";

}

cout << endl;

}

}

};

int main() {

int keys[6] = {23, 44, 12, 39, 33, 56};

int n = sizeof(keys) / sizeof(keys[0]);

cout << "Hash table of size 7 : \n";

HashTable ht2(7);

for (int i = 0; i < n; i++) ht2.insert(keys[i]);

ht2.display();

int searchKeys[3] = {12, 99, 44};

int pos;

cout << "\nSearch results:\n";

for (int key : searchKeys) {

if (ht2.search(key, pos)) {

cout << "Key " << key << " found at position " << pos << endl;

} else {

cout << "Key " << key << " NOT found in the hash table." << endl;

}

}

cout<<"Mehuli Sarkar, av.sc.u4cse24150";

return 0;

}

Output :

A screenshot of a computer code

AI-generated content may be incorrect.

3. Delete the keys {39, 23} from the hash table created in Q1. Show the new state of the table.

Code :

#include <iostream>

#include <vector>

#include <list>

using namespace std;

class HashTable {

private:

int size;

vector<list<int>> table;

int hashFunction(int key) {

return key % size;

}

public:

HashTable(int s) {

size = s;

table.resize(size);

}

void insert(int key) {

int index = hashFunction(key);

table[index].push\_back(key);

}

bool search(int key, int &index) {

index = hashFunction(key);

for (int k : table[index]) {

if (k == key) {

return true;

}

}

return false;

}

// Remove key from hash table

bool remove(int key) {

int index = hashFunction(key);

auto &chain = table[index];

for (auto it = chain.begin(); it != chain.end(); ++it) {

if (\*it == key) {

chain.erase(it);

return true; // key deleted

}

}

return false; // key not found

}

void display() {

for (int i = 0; i < size; i++) {

cout << i << "-->";

for (int key : table[i]) {

cout << key << " ";

}

cout << endl;

}

}

};

int main() {

int keys[6] = {23, 44, 12, 39, 33, 56};

int n = sizeof(keys) / sizeof(keys[0]);

cout << "Hash table of size 7 : \n";

HashTable ht2(7);

for (int i = 0; i < n; i++) ht2.insert(keys[i]);

ht2.display();

// Delete keys 39 and 23

cout << "\nDeleting keys 39 and 23...\n";

ht2.remove(39);

ht2.remove(23);

cout << "\nHash table after deletion:\n";

ht2.display();

cout<<”Mehuli Sarkar, AV.Sc.U4CSE24150”;

return 0;

}

Output :

A screenshot of a computer code

AI-generated content may be incorrect.

4. Compare the distribution of keys. Which gives better spread? Explain why.

When comparing the distribution of keys in hash tables of size 7 and size 10 using the keys {23, 44, 12, 39, 33, 56}, the hash table of size 10 provides a better spread. With size 7, the keys cluster more heavily in certain buckets; for example, keys 23 and 44 both map to bucket 2, and keys 12 and 33 both map to bucket 5, resulting in longer chains in those buckets and empty buckets elsewhere. On the other hand, with size 10, the keys are distributed more evenly across the buckets, with most buckets containing only one key and only one bucket having two keys. This reduces collisions and chain lengths, improving overall efficiency. Although using a prime number for the table size—like 7—is generally recommended to reduce clustering, the specific keys in this case happen to spread more evenly when the size is 10. Therefore, the hash table with size 10 demonstrates a better distribution of keys, leading to faster search, insertion, and deletion operations due to fewer collisions and shorter chains.