Experiment No-8

Vilakshan

Batch – B

Problem statement: Create a hash table and handle the collisions using linear probing with or without replacement.

Code:

#include <iostream>

#include <list>

using *namespace* std;

*class* HashTable

{

*int* capacity;

    list<*int*> \*table;

*public:*

    HashTable(*int* *V*);

*void* insertItem(*int* *key*, *int* *data*);

*void* deleteItem(*int* *key*);

*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;

    }

*int* getPrime(*int* *n*)

    {

        if (*n* % 2 == 0)

        {

*n*++;

        }

        while (!checkPrime(*n*))

        {

*n* += 2;

        }

        return *n*;

    }

*int* hashFunction(*int* *key*)

    {

        return (*key* % capacity);

    }

*void* displayHash();

};

HashTable::HashTable(*int* *c*)

{

*int* size = getPrime(*c*);

    this->capacity = size;

    table = new list<*int*>[capacity];

}

*void* HashTable::insertItem(*int* *key*, *int* *data*)

{

*int* index = hashFunction(*key*);

    table[index].push\_back(*data*);

}

*void* HashTable::deleteItem(*int* *key*)

{

*int* index = hashFunction(*key*);

    list<*int*>::iterator i;

    for (i = table[index].begin();

         i != table[index].end(); i++)

    {

        if (\*i == *key*)

            break;

    }

    if (i != table[index].end())

        table[index].erase(i);

}

*void* HashTable::displayHash()

{

    for (*int* i = 0; i < capacity; i++)

    {

        cout << "table[" << i << "]";

        for (*auto* x : table[i])

            cout << " --> " << x;

        cout << endl;

    }

}

*int* main()

{

*int* key[] = {231, 321, 212, 321, 433, 262};

*int* data[] = {123, 432, 523, 43, 423, 111};

*int* size = sizeof(key) / sizeof(key[0]);

    HashTable h(size);

    for (*int* i = 0; i < size; i++)

        h.insertItem(key[i], data[i]);

    h.deleteItem(12);

    h.displayHash();

}

OUTPUT:

