// Autogenerated with DRAKON Editor 1.31

const SIZE = 15;

function createNode(value, next = null) {

return {

Value: value,

Next: next,

};

}

function createHashTable(size) {

return {

Table: {},

Size: size,

};

}

function hashFunc(i, size) {

return i % size;

}

function hashInsert(hash, value) {

const index = hashFunc(value, hash.Size);

const element = createNode(value, hash.Table[index]);

hash.Table[index] = element;

return index;

}

function hashLookup(hash, value) {

const index = hashFunc(value, hash.Size);

if (hash.Table[index]) {

let t = hash.Table[index];

while (true) {

if (t) {

if (t.Value === value) {

console.log(value, "- is exist");

return;

}

t = t.Next;

} else {

break;

}

}

}

console.log("After removing ", value, "- does not exist");

return;

}

function hashLookup\_2(hash, value) {

const index = hashFunc(value, hash.Size);

if (hash.Table[index]) {

let t = hash.Table[index];

while (true) {

if (t) {

if (t.Value === value) {

console.log(value, "присутствует");

return;

}

t = t.Next;

} else {

break;

}

}

}

console.log(value, "отсутствует");

return;

}

function hashRemove(hash, value) {

const index = hashFunc(value, hash.Size);

if (!hash.Table[index]) {

return;

}

let node = hash.Table[index];

let nodePrev = null;

while (node) {

if (node.Value === value) {

if (nodePrev) {

nodePrev.Next = node.Next;

} else {

hash.Table[index] = node.Next;

}

return;

}

nodePrev = node;

node = node.Next;

}

}

function hashTravers(hash) {

for (const key in hash.Table) {

if (hash.Table[key]) {

let t = hash.Table[key];

while (t) {

console.log(" : ", t.Value);

t = t.Next;

}

}

console.log();

}

}

function main() {

const hash = createHashTable(SIZE);

for (let i = 0; i < 120; i++) {

hashInsert(hash, i);

}

hashLookup(hash, 74);

hashRemove(hash, 74);

hashLookup(hash, 74);

}

main();