```
#include <iostream>
#include<bits/stdc++.h>
using namespace std;

template<typename K,typename V>
class HashNode{
public:
    K key;
    V value;
    HashNode(K key,V value){
        this->key=key;
        this->value=value;
    }
};
template<typename K,typename V>
class HashMap{
    HashNode<K,V> **arr;
    int capacity,size;
    HashNode<K,V> *dummy;
public:
    HashMap(){
        capacity=20;
        size=0;
        arr= new HashNode<K,V>* [capacity];
        for(int i=0;i<capacity;i++){
            arr[i]= nullptr;
        }
        dummy=new HashNode<K,V>(-1,-1);
    }
    int HashCode(K key){
        return key%capacity;
    }
    void InsertNode(K key,V value){
        if(size>=capacity) return;
        HashNode<K,V>* newnode=new HashNode<K,V>(key,value);
        int NodeIndex=HashCode(key);
        while(arr[NodeIndex]!=nullptr && arr[NodeIndex]->key!=key &&
arr[NodeIndex]->key!=-1) {
            NodeIndex++;
            NodeIndex%=capacity;
        }
        if(arr[NodeIndex]==nullptr || arr[NodeIndex]->key==-1){ // don't renew
if key has already inserted
            arr[NodeIndex]=newnode;
            size++;
        }
    }
    V SearchNode(K key) {
        int NodeIndex = HashCode(key);
        int count = 0;
        while (arr[NodeIndex] != nullptr) {
            if (++count > capacity) return 0;
            if (arr[NodeIndex]->key == key) {
                return arr[NodeIndex]->value;
            }
            NodeIndex++;
            NodeIndex %= capacity;
        }
        return 0;
    }
    V DeleteNode(K key){
        int NodeIndex=HashCode(key);
        int count=0;
        while(arr[NodeIndex]!=nullptr){
            if(++count>capacity) return 0;
            if(arr[NodeIndex]->key==key){
                HashNode<K,V>* temp=arr[NodeIndex];
                arr[NodeIndex]=dummy;
                size--;
                return temp->value;
            }
            NodeIndex++;
            NodeIndex %= capacity;
        }
        return 0;
    }
    int sizeofMap()
    {
        return size;
    }

    //Return true if size is 0
    bool isEmpty()
    {
        return size == 0;
    }

    //Function to display the stored key value pairs
    void display()
    {
        for(int i=0 ; i<capacity ; i++)
        {
            if(arr[i] != nullptr && arr[i]->key != -1)
                cout <<i<< " key = " << arr[i]->key
                     <<"  value = "<< arr[i]->value << endl;
        }
    }
};

int main()
{
    HashMap<int, int> *h = new HashMap<int, int>;
    h->InsertNode(1,1);
    h->InsertNode(2,4);
    h->display();
    h->InsertNode(2,3);
    h->display();
    cout << h->sizeofMap() <<endl;
    cout << h->DeleteNode(2) << endl;
    h->display();
    cout << h->sizeofMap() <<endl;
    cout << h->isEmpty() << endl;
    cout << h->SearchNode(2);
    return 0;
}
```