**/\*WAP to implement hashing.\*/**

**#include<iostream>**

**#include<cstdlib>**

**#include<string>**

**#include<cstdio>**

**using namespace std;**

**const int T\_S = 200;**

**class HashTableEntry**

**{**

**public:**

**int k;**

**int v;**

**HashTableEntry(int k, int v)**

**{**

**this->k= k;**

**this->v = v;**

**}**

**};**

**class HashMapTable**

**{**

**private:**

**HashTableEntry \*\*t;**

**public:**

**HashMapTable()**

**{**

**t = new HashTableEntry \* [T\_S];**

**for (int i = 0; i< T\_S; i++)**

**{**

**t[i] = NULL;**

**}**

**}**

**int HashFunc(int k)**

**{**

**return k % T\_S;**

**}**

**void Insert(int k, int v)**

**{**

**int h = HashFunc(k);**

**while (t[h] != NULL && t[h]->k != k)**

**{**

**h = HashFunc(h + 1);**

**}**

**if (t[h] != NULL)**

**delete t[h];**

**t[h] = new HashTableEntry(k, v);**

**}**

**int SearchKey(int k)**

**{**

**int h = HashFunc(k);**

**while (t[h] != NULL && t[h]->k != k)**

**{**

**h = HashFunc(h + 1);**

**}**

**if (t[h] == NULL)**

**return -1;**

**else**

**return t[h]->v;**

**}**

**void Remove(int k)**

**{**

**int h = HashFunc(k);**

**while (t[h] != NULL)**

**{**

**if (t[h]->k == k)**

**break;**

**h = HashFunc(h + 1);**

**}**

**if (t[h] == NULL)**

**{**

**cout<<"No Element found at key "<<k<<endl;**

**return;**

**}**

**else**

**{**

**delete t[h];**

**}**

**cout<<"Element Deleted"<<endl;**

**}**

**~HashMapTable()**

**{**

**for (int i = 0; i < T\_S; i++)**

**{**

**if (t[i] != NULL)**

**delete t[i];**

**delete[] t;**

**}**

**}**

**};**

**int main()**

**{**

**HashMapTable hash;**

**int k, v;**

**int c;**

**while (1)**

**{**

**cout<<"1.Insert element into the table"<<endl;**

**cout<<"2.Search element from the key"<<endl;**

**cout<<"3.Delete element at a key"<<endl;**

**cout<<"4.Exit"<<endl;**

**cout<<"Enter your choice: ";**

**cin>>c;**

**switch(c)**

**{**

**case 1:**

**cout<<"Enter element to be inserted: ";**

**cin>>v;**

**cout<<"Enter key at which element to be inserted: ";**

**cin>>k;**

**hash.Insert(k, v);**

**break;**

**case 2:**

**cout<<"Enter key of the element to be searched: ";**

**cin>>k;**

**if (hash.SearchKey(k) == -1)**

**{**

**cout<<"No element found at key "<<k<<endl;**

**continue;**

**}**

**else**

**{**

**cout<<"Element at key "<<k<<" : ";**

**cout<<hash.SearchKey(k)<<endl;**

**}**

**break;**

**case 3:**

**cout<<"Enter key of the element to be deleted: ";**

**cin>>k;**

**hash.Remove(k);**

**break;**

**case 4:**

**exit(1);**

**default:**

**cout<<"\nEnter correct option\n";**

**}**

**}**

**return 0;**

**}**