```
/*Приведите содержимое хеш-таблицы, образованной вставками элементов с ключами
E A S Y Q U T I O N в указанном порядке в первоначально пустую таблицу из M = 5 списков
при использовании цепочек переполнения в виде неупорядоченных списков. Для
преобразования k-ой буквы алфавита в индекс таблицы используйте хеш-функцию 11k
mod M.*/

/*Отчет:
Был создан класс с реализацией хеш-таблицы. В нем присутствуют методы для получения ключа, получения значения,
установки значения, получения следующего элемента и его назначения.
Есть функция очистки всей таблицы, а также вывода ее на экран.
Функция hashFunc отвечает за возврат значения хеш-функции для заданного элемента.
Добавление в таблицу осуществляется функцией addToHashArray. Если элемента с данным ключом еще нет, то он добавляется в начало
списка. Если появляется коллизия, то элемент добавляется следующим элементом в цепочку переполнения.
В главной функции заданные заданием буквы добавляются в таблицу с помощью функции addToHashArray.*/

#include <iostream>

using namespace std;

class LinkedHashEntry {
private:
    int key;
    int value;
    LinkedHashEntry* next;
public:
    LinkedHashEntry(int key, int value) {
        this->key = key;
        this->value = value;
        this->next = NULL;
    }
    int getKey() {
        return key;
    }
    int getValue() {
        return value;
    }
    void setValue(int value) {
        this->value = value;
    }
    LinkedHashEntry* getNext() {
        return next;
    }
```

```
    void setNext(LinkedHashEntry* next) {
        this->next = next;
    }
};


void clearHashArray(LinkedHashEntry* hashArray[], int k) {
    for (int i = 0; i < k; i++) {
        hashArray[i] = NULL;
    }
}

void coutHashArray(LinkedHashEntry* hashArray[], int k) {
    for (int i = 0; i < k; i++) {
        LinkedHashEntry* iter = hashArray[i];
        while (iter != NULL) {
            cout << " " << iter->getKey() << " " << iter->getValue()
<< "(" << (char)iter->getValue() << ")  ";
            iter = iter->getNext();
        }
        cout << "  NULL\n";
    }
}

int hashFunc(int value, int M) {
    return 11 * value % M;
}

void addToHashArray(LinkedHashEntry* hashArray[], int M, int value,
int len[]) {
    int key = hashFunc(value, M);
    len[key]++;
    LinkedHashEntry* obj = new LinkedHashEntry(key, value);
    if (hashArray[key] == NULL) {
        hashArray[key] = obj;
        return;
    }
    LinkedHashEntry* iter = hashArray[key];
    while (iter->getNext() != NULL) {
        iter = iter->getNext();
    }
    iter->setNext(obj);
}


int main() {
    // E A S Y Q U T I O N
    setlocale(NULL, "");
    const int M = 5;
    cout << "\nРазмер хеш-таблицы: " << M << endl;
    LinkedHashEntry* hashArray[M];
```

```
    int lenOfHashArrayChains[M];
    for (int j = 0; j < M; j++)
        lenOfHashArrayChains[j] = 0;
    clearHashArray(hashArray, M);
    addToHashArray(hashArray, M, (int)'E', lenOfHashArrayChains);
    addToHashArray(hashArray, M, (int)'A', lenOfHashArrayChains);
    addToHashArray(hashArray, M, (int)'S', lenOfHashArrayChains);
    addToHashArray(hashArray, M, (int)'Y', lenOfHashArrayChains);
    addToHashArray(hashArray, M, (int)'Q', lenOfHashArrayChains);
    addToHashArray(hashArray, M, (int)'U', lenOfHashArrayChains);
    addToHashArray(hashArray, M, (int)'T', lenOfHashArrayChains);
    addToHashArray(hashArray, M, (int)'I', lenOfHashArrayChains);
    addToHashArray(hashArray, M, (int)'O', lenOfHashArrayChains);
    addToHashArray(hashArray, M, (int)'N', lenOfHashArrayChains);
    coutHashArray(hashArray, M);
    return 0;
}
```