Задача B. Map (2 балла)

Я использовал технику обработки столкновений: Separate chaining (open hashing)

map.in	map.out
put hello privet put bye poka get hello get bye delete hello get hello	privet poka none

```
struct node {
    string key;
    string data;
    node* next;
};
```

-> Я создаю структуру для хранения элементов

```
node* hTable[Size] = {nullptr};
```

Я создал хеш-таблицу и инициализировал ее NULL значением

```
int hash_key (string key){
    int k = 0;
    for (int i=0; i<key.length(); i++) {
        k += ((int)key[i])*(i+1);
    }return k%97;
}
```

-> это моя хеш-функция

- Я следую кодировке ASCII для каждого символа, умножаю на индекс + 1, а затем складываю затем разделите остаток его на наименьшую возможную строку (в этом случае “a”)

```
node* search_key(string key){
    int k = hash_key(key);
    node* p = hTable[k];
    while (p!=nullptr && p->key != key) {
        p = p->next;
    }
    if (p == nullptr) {
        return nullptr;
    }return p;
}
```

-> это функция для поиска : Мы используем хеш-функцию для доступа к соответствующему bucket, чтобы найти

Мы создаем указатели и проходим каждый элемент связанного списка для поиска

```
void put_command(string key, string data){
    node* p = search_key(key);
    int k = hash_key(key);
    if (p!=nullptr) {
        p->data = data;
        return;
    }else{
        node* p1 = new node();
        p1->key = key;
        p1->data = data;
        p1->next = nullptr;
        if (hTable[k] == nullptr) {
            hTable[k] = p1;
        }else{
            p1->next = hTable[k];
            hTable[k] = p1;
        }return;
    }
}
```

-> Вот функция для добавления: Мы используем хеш-функцию для доступа к bucket для добавления

- Если он находит существующий ключ, измените его значение

- в противном случае мы создадим новый узел с его ключевым словом и значением
+ Если A в ключе k пусто, добавьте вновь созданный узел
+ в противном случае они добавляют его в начало односвязного списка

```
void delete_command(string key){
    int k = hash_key(key);
    node* p = hTable[k];
    node* p1 = nullptr;
    while (p != nullptr && p->key != key) {
        p1 = p;
        p = p->next;
    }
    if (p == nullptr) {
        return;
    }
    else if (p == hTable[k]){
        hTable[k] = hTable[k]->next;
    }else
        p1->next = p->next;
}
```

-> Это функция, используемая для удаления

- Мы используем хеш-функцию для доступа к bucket, содержащей удаляемый элемент.

- мы просматриваем единый связанный список, чтобы найти элемент, который нужно удалить, а затем удаляем элемент из единого связанного списка.