Задача C. LinkedMap (2 балла)

linkedmap.in	linkedmap.out
put zero a put one b put two c put three d put four e get two prev two next two delete one delete three get two prev two next two next four	c b d c a e none

Метод этой задачи такой же, как и для задачи B, но нужно только добавить массив string, чтобы эффективно выполнять команды prev и next, кроме того, мы также можем использовать список с двойной связью (double linked list) для выполнения этой задачи.

```
node* H_table[TABLESIZE] = {nullptr};
string value_array[100003];
int counter = 0;
```

- создает хеш-таблицу и инициализирует ее значением NULL
- создать массив string, count переменных для управления количеством элементов массива

```
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”)

```
astring get_command(string key) {
    node* X;
    int k = hash_function(key);
    X = H_table[k];
    if (!X) {
        return "";
    }
    if (X->key == key) {
        return X->value;
    }
    while (X->next) {
        if (X->key == key)
            return X->value;
        X = X->next;
    }
    if (X->key == key) {
        return X->value;
    }
    else {
        return "";
    }
}
```

Мы используем хеш-функцию для доступа к соответствующему bucket, чтобы найти

мы также ищем каждый элемент односвязного списка тобы найти

```
avoid put_command(string key, string value){
   int k = hash_function(key);
   node* X = new node();
   X->key = key;
   X->value = value;
   X->next = nullptr;
   if (H_table[k] == nullptr) {
      H_table[k] = X;
      H_table[k]->index = counter;
      value_array[counter] = value;
      counter++;
   }
   else {
      node* X1;
      X1 = H_table[k];
      while (X1->next) {
         if (X1->key == key) {
            X1->value = value;
            value_array[X1->index] = value;
            return;
         }
         X1 = X1->next;
      }

      if (X1->key != key) {
         X1->next = X;
         X->index = counter;
         value_array[counter] = value;
         counter++;
       }
       else {
         X1->value = value;
         value_array[X1->index] = value;
      }
   }
}
```

- Используйте хеш-функцию, чтобы перейти к соответствующему сегменту для добавления

- создать новый узел, ключевое слово и значение

- Если bucket пуста, добавьте узел и добавьте значение этого узла в массив string.

- Напротив выполняет просмотр каждого элемента в односвязном списке

  + Если существует ключ, измените его значение и измените значение ключа в массиве string

  + Если ключ не существует, добавьте узел X в конец односвязного списка и добавьте массив string

  + Если последний элемент в односвязном списке уже имеет ключ, измените его значение и измените значение ключа в массиве string

```
void delete_command(string key) {
   node* X;
   int index_value = hash_function(key);
   X = H_table[index_value];
   if (X == nullptr)
      return;
   if (X->key == key) {
      value_array[H_table[index_value]->index] =
"";
      H_table[index_value] = X->next;
      return;
   }
   while (X->next) {
      if (X->next->key == key) {
         value_array[X->next->index] = "";
         X->next = X->next->next;
         return;
      }
      X = X->next;
   }
```

- Используйте хеш-функцию, чтобы перейти к соответствующему сегменту для удалить

- если bucket = NULL не нужно ничего делать

- выполнить просмотр и удаление в единственном связанном списке и удаление в массиве string

```
string next_command(string key) {
    int k = hash_function(key);
    node* X = H_table[k];
    if (!X) {
        return "";
    }
    if (X->key == key) {
        for (int i = X->index + 1; i < counter; i++) {
            if (value_array[i] != "") {
                return value_array[i];
            }
        }
        return "";
    }
    while (X->next) {
        if (X->key == key) {
            for (int i = X->index + 1; i < counter; i++) {
                if (value_array[i] != "") {
                    return value_array[i];
                }
            }
            return "";
        }
        X = X->next;
    }
    if (X->key == key) {
        for (int i = X->index + 1; i < 100000; i++) {
            if (value_array[i] != "") {
                return value_array[i];
            }
        }
        return "";
    }
    else {
        return "";
    }
}
```

- использует хеш-функцию для перехода в bucket k ведро

- если bucket k пусто, return “none”

- напротив:

  + Если вы найдете правильный ключ, перемещается вправо, поскольку элемент содержит соответствующий ключ, чтобы найти next элемент.

  + Если соответствующий ключ не найден, просмотрите односвязный список, чтобы найти правильный ключ, затем просмотрите массив string вправо, поскольку элемент содержит соответствующий ключ, чтобы найти next

  + Если последний элемент содержит соответствующий ключ, мы также прокручиваем вправо, чтобы найти next

  + в противном случае return “none”

```
string prev_command(string key_value) {
    int index_value = hash_function(key_value);
    node* X = H_table[index_value];
    if (!X) {
        return "";
    }
    if (X->key == key_value) {
        for (int i = X->index - 1; i >= 0; i--) {
            if (value_array[i] != "") {
                return value_array[i];
            }
        }
        return "";
    }
    while (X->next) {
        if (X->key == key_value) {
            for (int i = X->index - 1; i >= 0; i--) {
                if (value_array[i] != "") {
                    return value_array[i];
                }
            }
            return "";
        }
        X = X->next;
    }
    if (X->key == key_value) {
        for (int i = X->index - 1; i >= 0; i--) {
            if (value_array[i] != "") {
                return value_array[i];
            }
        }
        return "";
    }
    else {
        return "";
    }
}
```

Найти prev аналогичен найти next, но будет проходить слева от массива string