

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«Белгородский государственный технологический университет им.
В.Г. Шухова»**

(БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной техники и
автоматизированных систем

Лабораторная работа №13

По дисциплине: «Основы программирования»

Тема: «Множества»

Выполнил: студент группы ВТ-231

Борченко Александр Сергеевич

Проверили:

Черников Сергей Викторович

Новожен Никита Викторович

Белгород 2023

Цель работы: закрепление навыков работы со структурами, изучение простых способов представления множеств в памяти ЭВМ.

Содержание работы:

Задание №1. Выполнить реализацию множества на типе <code>uint32_t</code> . Содержимое файла <i>bitset.h</i>	3
Задание №2. На неупорядоченном массиве.....	6
Задание №3. * На упорядоченном массиве	14

Задачи с code forces

Задание №1. Определи маршрут (1056A).....	20
Задание №2. Пропущенная серия (440A)	22
Задание №3. Перестановка букв (1093B).....	23
Задание №4. Тихий класс (1166A).....	25
Задание №5. Щедрый Кефа (841A)	26
Задание №6. Перекраска собачек (1025A).....	27
Задание №7. Ступени (1011A)	28
Задание №8. Башни (37A)	30
Задание №9. Бейджик (1020B).....	32
Задание №10. Разнообразие – это хорошо (672B)	33
Задание №11. Игра: Банковские карты (777B).....	34
Задание №12. Противоположности притягиваются (131B).....	36

Задание №1. Выполнить реализацию множества на типе `uint32_t`.
Содержимое файла *bitset.h*.

```
#ifndef БИБЛИОТЕКИ_BIT_SET_H
#define БИБЛИОТЕКИ_BIT_SET_H
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>
#include <assert.h>
#include <malloc.h>

typedef struct bitset {
    uint32_t *values;
    uint32_t max_value;
} bitset;

bitset bitset_create(unsigned max_value) {
    bitset set;
    set.max_value = max_value;

    uint32_t array_size = (max_value + 31) / 32;

    set.values = (uint32_t *)calloc(array_size, sizeof(uint32_t));

    return set;
}

bool bitset_in(bitset set, uint32_t value) {
    if (value > set.max_value) {
        return false;
    }
    uint32_t index = value / 32;
    uint32_t bit_offset = value % 32;

    return (set.values[index] & (1 << bit_offset)) != 0;
}

void bitset_insert(bitset *set, uint32_t value) {
    if (value > set->max_value) {
        return;
    }

    uint32_t index = value / 32;
    uint32_t bit_offset = value % 32;

    set->values[index] |= (1 << bit_offset);
}

bool bitset_isEqual(bitset set1, bitset set2) {
    if (set1.max_value != set2.max_value) {
        return false;
    }

    uint32_t array_size = (set1.max_value + 31) / 32;
    for (uint32_t i = 0; i < array_size; ++i) {
        if (set1.values[i] != set2.values[i]) {
            return false;
        }
    }
    return true;
}
```

```

bool bitset_isSubset(bitset set1, bitset set2) {
    if (set1.maxValue > set2.maxValue) {
        return false;
    }

    uint32_t arraySize = (set1.maxValue + 31) / 32;
    for (uint32_t i = 0; i < arraySize; ++i) {
        if ((set1.values[i] & ~set2.values[i]) != 0) {
            return false;
        }
    }
    return true;
}

void bitset_deleteElements(bitset *set) {
    free(set->values);
    set->values = NULL;
}

bitset bitset_union(bitset set1, bitset set2) {
    bitset result = bitset_create(set1.maxValue > set2.maxValue ?
set1.maxValue : set2.maxValue);

    for (uint32_t i = 0; i < result.maxValue; ++i) {
        if (bitset_in(set1, i) || bitset_in(set2, i)) {
            bitset_insert(&result, i);
        }
    }

    return result;
}

bitset bitset_intersection(bitset set1, bitset set2) {
    bitset result = bitset_create(set1.maxValue < set2.maxValue ?
set1.maxValue : set2.maxValue);

    for (uint32_t i = 0; i < result.maxValue; ++i) {
        if (bitset_in(set1, i) && bitset_in(set2, i)) {
            bitset_insert(&result, i);
        }
    }

    return result;
}

bitset bitset_difference(bitset set1, bitset set2) {
    bitset result = bitset_create(set1.maxValue);

    for (uint32_t i = 0; i < set1.maxValue; ++i) {
        if (bitset_in(set1, i) && !bitset_in(set2, i)) {
            bitset_insert(&result, i);
        }
    }

    return result;
}

bitset bitset_symmetricDifference(bitset set1, bitset set2) {
    bitset result = bitset_create(set1.maxValue > set2.maxValue ?
set1.maxValue : set2.maxValue);

    for (uint32_t i = 0; i < result.maxValue; ++i) {
        if ((bitset_in(set1, i) && !bitset_in(set2, i)) || (!bitset_in(set1,

```

```

i) && bitset_in(set2, i))) {
    bitset_insert(&result, i);
}

return result;
}

bitset bitset_complement(bitset set) {
    bitset result = bitset_create(set.maxValue);

    for (uint32_t i = 0; i < set.maxValue; ++i) {
        if (!bitset_in(set, i)) {
            bitset_insert(&result, i);
        }
    }

    return result;
}

void bitset_print(bitset set) {
    printf("{ ");
    for (uint32_t i = 0; i <= set.maxValue; ++i) {
        if (bitset_in(set, i)) {
            printf("%u ", i);
        }
    }
    printf("}\n");
}

#endif //БИБЛИОТЕКИ_BIT_SET_H

```

Задание №2. На неупорядоченном массиве

```
#ifndef C_UNORDERED_SET_H
#define C_UNORDERED_SET_H
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>
#include <assert.h>
#include <malloc.h>

typedef struct unordered_array_set {
    int *data;
    size_t size;
    size_t capacity;
} unordered_array_set;

unordered_array_set unordered_array_set_create(size_t capacity) {
    unordered_array_set set;
    set.data = (int *)malloc(capacity * sizeof(int));
    set.size = 0;
    set.capacity = capacity;
    return set;
}

/*size_t unordered_array_set_in(unordered_array_set set, int value) {
    for (size_t i = 0; i < set.size; i++) {
        if (set.data[i] == value) {
            return i;
        }
    }
    return set.size;
}
*/

size_t linearSearch(const int a[], const size_t n, int x) {
    for (size_t i = 0; i < n; i++)
        if (a[i] == x)
            return i;
    return n;
}

size_t unordered_array_set_in(unordered_array_set* set, int value) {
    return linearSearch(set->data, set->size, value);
}

bool unordered_array_set_isSubset(unordered_array_set subset,
    unordered_array_set set) {
    for (size_t i = 0; i < subset.size; i++) {
        bool found = false;
        for (size_t j = 0; j < set.size; ++j) {
            if (subset.data[i] == set.data[j]) {
                found = true;
                break;
            }
        }

        return found;
    }
}

int compare(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
}
```

```

bool unordered_array_set_isEqual(unordered_array_set set1,
unordered_array_set set2) {
    if (set1.size != set2.size) {
        return false;
    }

    qsort(set1.data, set1.size, sizeof(int), compare);
    qsort(set2.data, set2.size, sizeof(int), compare);

    for (size_t i = 0; i < set1.size; i++) {
        if (set1.data[i] != set2.data[i]) {
            return false;
        }
    }
    return true;
}

void unordered_array_set_isAbleAppend(unordered_array_set *set) {
    assert(set->size >= set->capacity);
}

void unordered_array_set_insert(unordered_array_set *set, int value) {
    if (set->size >= set->capacity) {
        size_t new_capacity = (set->capacity == 0) ? 1 : set->capacity * 2;
        int *new_data = realloc(set->data, new_capacity * sizeof(int));

        set->data = new_data;
        set->capacity = new_capacity;
    }

    set->data[set->size++] = value;
}

unordered_array_set unordered_array_set_create_from_array(const int *a,
size_t size) {
    unordered_array_set set = {NULL, 0, 0};
    for (size_t i = 0; i < size; ++i) {
        unordered_array_set_insert(&set, a[i]);
    }
    return set;
}

void unordered_array_set_deleteElement(unordered_array_set *set, int value) {
    for (size_t i = 0; i < set->size; ++i) {
        if (set->data[i] == value) {
            for (size_t j = i; j < set->size - 1; ++j) {
                set->data[j] = set->data[j + 1];
            }
            set->size--;
            return;
        }
    }
}

unordered_array_set unordered_array_set_union(unordered_array_set set1,
unordered_array_set set2) {
    unordered_array_set result = {NULL, 0, 0};

    for (size_t i = 0; i < set1.size; i++) {
        if (unordered_array_set_in(result, set1.data[i]) == result.size) {
            unordered_array_set_insert(&result, set1.data[i]);
        }
    }
}

```

```

    }

    for (size_t i = 0; i < set2.size; i++) {
        if (unordered_array_set_in(result, set2.data[i]) == result.size) {
            unordered_array_set_insert(&result, set2.data[i]);
        }
    }

    return result;
}

unordered_array_set unordered_array_set_intersection(unordered_array_set
set1, unordered_array_set set2) {
    unordered_array_set intersection = {NULL, 0, 0};
    for (size_t i = 0; i < set1.size; i++) {
        if (unordered_array_set_in(set2, set1.data[i]) != set1.size) {
            unordered_array_set_insert(&intersection, set1.data[i]);
        }
    }
    return intersection;
}

unordered_array_set unordered_array_set_difference(unordered_array_set set1,
unordered_array_set set2) {
    unordered_array_set difference = {NULL, 0, 0};
    for (size_t i = 0; i < set1.size; i++) {
        if (unordered_array_set_in(set2, set1.data[i]) == set1.size) {
            unordered_array_set_insert(&difference, set1.data[i]);
        }
    }
    return difference;
}

unordered_array_set unordered_array_set_complement(unordered_array_set set,
unordered_array_set universumSet) {
    unordered_array_set complement = {NULL, 0, 0};
    for (size_t i = 0; i < universumSet.size; i++) {
        if (unordered_array_set_in(set, universumSet.data[i]) == set.size) {
            unordered_array_set_insert(&complement, universumSet.data[i]);
        }
    }
    return complement;
}

void unordered_array_set_delete(unordered_array_set set) {
    free(set.data);
}

unordered_array_set
unordered_array_set_symmetricDifference(unordered_array_set set1,
unordered_array_set set2) {
    unordered_array_set diff1 = unordered_array_set_difference(set1, set2);
    unordered_array_set diff2 = unordered_array_set_difference(set2, set1);
    unordered_array_set symmetric_difference =
unordered_array_set_union(diff1, diff2);
    unordered_array_set_delete(diff1);
    unordered_array_set_delete(diff2);
    return symmetric_difference;
}

void unordered_array_set_print(unordered_array_set set) {
    printf("{ ");
    for (size_t i = 0; i < set.size; ++i) {

```



```
        printf("%d ", set.data[i]);  
    }  
    printf("}\n");  
}  
  
#endif //C_UNORDERED_SET_H
```

Тесты:

```
#include "unordered_set.h"

void test_unordered_array_set_union1() {
    unordered_array_set set1 = unordered_array_set_create_from_array((int[])
{1, 2}, 2);
    unordered_array_set set2 = unordered_array_set_create_from_array((int[])
{2, 3}, 2);
    unordered_array_set resSet = unordered_array_set_union(set1, set2);
    unordered_array_set expectedSet =
unordered_array_set_create_from_array((int[]) {1, 2, 3}, 3);
    assert(unordered_array_set_isEqual(resSet, expectedSet));
    unordered_array_set_delete(set1);
    unordered_array_set_delete(set2);
    unordered_array_set_delete(resSet);
    unordered_array_set_delete(expectedSet);
}

void test_unordered_array_set_union2() {
    unordered_array_set set1 = unordered_array_set_create_from_array((int[])
{1, 2, 3}, 3);
    unordered_array_set set2 = unordered_array_set_create_from_array((int[])
{}, 0);
    unordered_array_set resSet = unordered_array_set_union(set1, set2);
    unordered_array_set expectedSet =
unordered_array_set_create_from_array((int[]) {1, 2, 3}, 3);
    assert(unordered_array_set_isEqual(resSet, expectedSet));
    unordered_array_set_delete(set1);
    unordered_array_set_delete(set2);
    unordered_array_set_delete(resSet);
    unordered_array_set_delete(expectedSet);
}

void test_unordered_array_set_union() {
    test_unordered_array_set_union1();
    test_unordered_array_set_union2();
}

void test_unordered_array_set_in1() {
    unordered_array_set set = {NULL, 0, 0};
    unordered_array_set_insert(&set, 5);
    unordered_array_set_insert(&set, 10);
    size_t position = unordered_array_set_in(set, 10);

    assert(position == 1);
    unordered_array_set_delete(set);
}

void test_unordered_array_set_in2() {
    unordered_array_set set = {NULL, 0, 0};
    unordered_array_set_insert(&set, 5);
    unordered_array_set_insert(&set, 10);
    size_t position = unordered_array_set_in(set, 15);

    assert(position == set.size);
    unordered_array_set_delete(set);
}

void test_unordered_array_set_in() {
    test_unordered_array_set_in1();
    test_unordered_array_set_in2();
}
```

```

void test_unordered_array_set_isSubset1() {
    unordered_array_set emptySet = {NULL, 0, 0};
    unordered_array_set set = {NULL, 0, 0};
    unordered_array_set_insert(&set, 5);

    assert(!unordered_array_set_isSubset(emptySet, set));
}

void test_unordered_array_set_isSubset2() {
    unordered_array_set set = {NULL, 0, 0};
    unordered_array_set_insert(&set, 5);
    unordered_array_set_insert(&set, 10);

    assert(unordered_array_set_isSubset(set, set));
}

void test_unordered_array_set_isSubset3() {
    unordered_array_set set1 = {NULL, 0, 0};
    unordered_array_set set2 = {NULL, 0, 0};
    unordered_array_set_insert(&set1, 5);
    unordered_array_set_insert(&set2, 10);

    assert(!unordered_array_set_isSubset(set1, set2));
}

void test_unordered_array_set_isSubset() {
    test_unordered_array_set_isSubset1();
    test_unordered_array_set_isSubset2();
    test_unordered_array_set_isSubset3();
}

void test_unordered_array_set_insert() {
    unordered_array_set set = {NULL, 0, 0};
    unordered_array_set_insert(&set, 5);

    assert(set.size == 1 && set.data[0] == 5);
    unordered_array_set_delete(set);
}

void test_unordered_array_set_deleteElement() {
    unordered_array_set set = {NULL, 0, 0};
    unordered_array_set_insert(&set, 5);
    unordered_array_set_insert(&set, 10);
    unordered_array_set_deleteElement(&set, 5);

    assert(set.size == 1 && set.data[0] == 10);

    unordered_array_set_delete(set);
}

void test_unordered_array_set_intersection() {
    unordered_array_set set1 = {NULL, 0, 0};
    unordered_array_set set2 = {NULL, 0, 0};
    unordered_array_set_insert(&set1, 5);
    unordered_array_set_insert(&set1, 10);
    unordered_array_set_insert(&set2, 10);
    unordered_array_set_insert(&set2, 15);
    unordered_array_set intersection = unordered_array_set_intersection(set1,
set2);

    assert(intersection.size == 1 && intersection.data[0] == 10);
}

```

```

        unordered_array_set_delete(set1);
        unordered_array_set_delete(set2);
        unordered_array_set_delete(intersection);
    }

void test_unordered_array_set_difference() {
    unordered_array_set set1 = {NULL, 0, 0};
    unordered_array_set set2 = {NULL, 0, 0};
    unordered_array_set_insert(&set1, 5);
    unordered_array_set_insert(&set1, 10);
    unordered_array_set_insert(&set2, 10);
    unordered_array_set_insert(&set2, 15);
    unordered_array_set difference = unordered_array_set_difference(set1,
set2);

    assert(difference.size == 1 && difference.data[0] == 5);

    unordered_array_set_delete(set1);
    unordered_array_set_delete(set2);
    unordered_array_set_delete(difference);
}

void test_unordered_array_set_symmetricDifference() {
    unordered_array_set set1 = {NULL, 0, 0};
    unordered_array_set set2 = {NULL, 0, 0};
    unordered_array_set_insert(&set1, 5);
    unordered_array_set_insert(&set1, 10);
    unordered_array_set_insert(&set2, 10);
    unordered_array_set_insert(&set2, 15);
    unordered_array_set symmetric_difference =
unordered_array_set_symmetricDifference(set1, set2);

    assert(symmetric_difference.size == 2 && symmetric_difference.data[0] ==
5 && symmetric_difference.data[1] == 15);

    unordered_array_set_delete(set1);
    unordered_array_set_delete(set2);
    unordered_array_set_delete(symmetric_difference);
}

void test_unordered_array_set_complement() {
    unordered_array_set set = {NULL, 0, 0};
    unordered_array_set_insert(&set, 5);
    unordered_array_set_insert(&set, 10);
    unordered_array_set universumSet = {NULL, 0, 0};
    unordered_array_set_insert(&universumSet, 5);
    unordered_array_set_insert(&universumSet, 10);
    unordered_array_set_insert(&universumSet, 15);
    unordered_array_set complement = unordered_array_set_complement(set,
universumSet);

    assert(complement.size == 1 && complement.data[0] == 15);

    unordered_array_set_delete(set);
    unordered_array_set_delete(universumSet);
    unordered_array_set_delete(complement);
}

void test() {
    test_unordered_array_set_in();
    test_unordered_array_set_isSubset();
    test_unordered_array_set_insert();
}

```

```
test_unordered_array_set_union();
test_unordered_array_set_deleteElement();
test_unordered_array_set_intersection();
test_unordered_array_set_difference();
test_unordered_array_set_symmetricDifference();
test_unordered_array_set_complement();
}

int main() {
    test();

    return 0;
}
```

Задание №3. * На упорядоченном массиве

```
#ifndef БИБЛИОТЕКИ_ORDERED_SET_H
#define БИБЛИОТЕКИ_ORDERED_SET_H
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>
#include <assert.h>
#include <malloc.h>
#include "array.h"

typedef struct ordered_array_set {
    int *data;
    size_t size;
    size_t capacity;
} ordered_array_set;

ordered_array_set ordered_array_set_create(size_t capacity) {
    ordered_array_set set;
    set.data = (int *) malloc(capacity * sizeof(int));
    set.size = 0;
    set.capacity = capacity;
    return set;
}

ordered_array_set ordered_array_set_create_from_array(const int *a, size_t
size) {
    ordered_array_set set;
    set.data = (int *) malloc(size * sizeof(int));
    for (size_t i = 0; i < size; ++i) {
        set.data[i] = a[i];
    }
    set.size = size;
    set.capacity = size;
    return set;
}

size_t ordered_array_set_in(ordered_array_set *set, int value) {
    for (size_t i = 0; i < set->size; ++i) {
        if (set->data[i] == value) {
            return i;
        }
    }
    return set->size;
}

bool ordered_array_set_isEqual(ordered_array_set set1, ordered_array_set
set2) {
    if (set1.size != set2.size) {
        return false;
    }
    for (size_t i = 0; i < set1.size; ++i) {
        if (set1.data[i] != set2.data[i]) {
            return false;
        }
    }
    return true;
}

bool ordered_array_set_isSubset(ordered_array_set subset, ordered_array_set
set) {
    for (size_t i = 0; i < subset.size; ++i) {
        if (ordered_array_set_in(&set, subset.data[i]) == set.size) {
```

```

        return false;
    }
}
return true;
}

void ordered_array_set_isAbleAppend(ordered_array_set *set) {
    assert(set->size >= set->capacity);
}

void ordered_array_set_insert(ordered_array_set *set, int value) {
    size_t i;
    for (i = set->size; i > 0 && set->data[i - 1] > value; --i) {
        set->data[i] = set->data[i - 1];
    }
    set->data[i] = value;
    ++set->size;
}

void ordered_array_set_deleteElement(ordered_array_set *set, int value) {
    size_t i = ordered_array_set_in(set, value);
    if (i < set->size) {
        for (; i < set->size - 1; ++i) {
            set->data[i] = set->data[i + 1];
        }
        set->size--;
    }
}

ordered_array_set ordered_array_set_union(ordered_array_set set1,
ordered_array_set set2) {
    ordered_array_set result = ordered_array_set_create(set1.size +
set2.size);
    size_t i = 0, j = 0, k = 0;
    while (i < set1.size && j < set2.size) {
        if (set1.data[i] < set2.data[j]) {
            result.data[k++] = set1.data[i++];
        } else if (set2.data[j] < set1.data[i]) {
            result.data[k++] = set2.data[j++];
        } else {
            result.data[k++] = set1.data[i++];
            ++j;
        }
    }
    while (i < set1.size) {
        result.data[k++] = set1.data[i++];
    }
    while (j < set2.size) {
        result.data[k++] = set2.data[j++];
    }
    result.size = k;
    return result;
}

ordered_array_set ordered_array_set_intersection(ordered_array_set set1,
ordered_array_set set2) {
    ordered_array_set result = ordered_array_set_create(set1.size);
    size_t i = 0, j = 0, k = 0;
    while (i < set1.size && j < set2.size) {
        if (set1.data[i] < set2.data[j]) {
            i++;
        } else if (set2.data[j] < set1.data[i]) {
            j++;
        } else {
            result.data[k++] = set1.data[i];
            i++;
            j++;
        }
    }
    result.size = k;
    return result;
}

```

```

        } else {
            result.data[k++] = set1.data[i++];
            j++;
        }
    }
    result.size = k;
    return result;
}

ordered_array_set ordered_array_set_difference(ordered_array_set set1,
ordered_array_set set2) {
    ordered_array_set result = ordered_array_set_create(set1.size);
    size_t i = 0, j = 0, k = 0;
    while (i < set1.size && j < set2.size) {
        if (set1.data[i] < set2.data[j]) {
            result.data[k++] = set1.data[i++];
        } else if (set2.data[j] < set1.data[i]) {
            j++;
        } else {
            i++;
            j++;
        }
    }
    while (i < set1.size) {
        result.data[k++] = set1.data[i++];
    }
    result.size = k;
    return result;
}

ordered_array_set ordered_array_set_symmetricDifference(ordered_array_set
set1, ordered_array_set set2) {
    ordered_array_set result1 = ordered_array_set_difference(set1, set2);
    ordered_array_set result2 = ordered_array_set_difference(set2, set1);
    ordered_array_set result = ordered_array_set_union(result1, result2);
    free(result1.data);
    free(result2.data);
    return result;
}

ordered_array_set ordered_array_set_complement(ordered_array_set set,
ordered_array_set universumSet) {
    ordered_array_set result = ordered_array_set_difference(universumSet,
set);
    return result;
}

void ordered_array_set_print(ordered_array_set set) {
    printf("{ ");
    for (size_t i = 0; i < set.size; ++i) {
        printf("%d ", set.data[i]);
    }
    printf("}\n");
}

void ordered_array_set_delete(ordered_array_set set) {
    free(set.data);
}

#endif //БИБЛИОТЕКИ_ORDERED_SET_H

```


Тесты:

```
#include <assert.h>
#include "ordered_set.h"

void test_ordered_array_set_in() {
    ordered_array_set set = ordered_array_set_create_from_array((int[]) {1,
2, 3, 4, 5}, 5);
    assert(ordered_array_set_in(&set, 3) == 2);
    assert(ordered_array_set_in(&set, 6) == set.size);
    ordered_array_set_delete(set);
}

void test_ordered_array_set_isSubset() {
    ordered_array_set set1 = ordered_array_set_create_from_array((int[]) {1,
2, 3}, 3);
    ordered_array_set set2 = ordered_array_set_create_from_array((int[]) {1,
2, 3, 4, 5}, 5);
    ordered_array_set set3 = ordered_array_set_create_from_array((int[]) {1,
2, 6}, 3);
    assert(ordered_array_set_isSubset(set1, set2) == true);
    assert(ordered_array_set_isSubset(set2, set1) == false);
    assert(ordered_array_set_isSubset(set1, set3) == false);
    ordered_array_set_delete(set1);
    ordered_array_set_delete(set2);
    ordered_array_set_delete(set3);
}

void test_ordered_array_set_insert() {
    ordered_array_set set = ordered_array_set_create(5);
    ordered_array_set_insert(&set, 3);
    ordered_array_set_insert(&set, 1);
    ordered_array_set_insert(&set, 4);
    ordered_array_set_insert(&set, 2);
    ordered_array_set_insert(&set, 5);
    assert(ordered_array_set_in(&set, 3) == 2);
    assert(ordered_array_set_in(&set, 6) == set.size);
    ordered_array_set_delete(set);
}

void test_ordered_array_set_union() {
    ordered_array_set set1 = ordered_array_set_create_from_array((int[]) {1,
2, 4}, 3);
    ordered_array_set set2 = ordered_array_set_create_from_array((int[]) {2,
3, 5}, 3);
    ordered_array_set result = ordered_array_set_union(set1, set2);
    ordered_array_set expectedSet =
ordered_array_set_create_from_array((int[]) {1, 2, 3, 4, 5}, 5);
    assert(ordered_array_set_isEqual(result, expectedSet) == true);
    ordered_array_set_delete(set1);
    ordered_array_set_delete(set2);
    ordered_array_set_delete(result);
    ordered_array_set_delete(expectedSet);
}

void test_ordered_array_set_deleteElement() {
    ordered_array_set set = ordered_array_set_create_from_array((int[]) {1,
2, 3, 4, 5}, 5);
    ordered_array_set_deleteElement(&set, 3);
    ordered_array_set expectedSet =
ordered_array_set_create_from_array((int[]) {1, 2, 4, 5}, 4);
    assert(ordered_array_set_isEqual(set, expectedSet) == true);
    ordered_array_set_delete(set);
}
```

```

        ordered_array_set_delete(expectedSet);
    }

void test_ordered_array_set_intersection() {
    ordered_array_set set1 = ordered_array_set_create_from_array((int[]) {1,
2, 3, 4}, 4);
    ordered_array_set set2 = ordered_array_set_create_from_array((int[]) {2,
3, 5}, 3);
    ordered_array_set result = ordered_array_set_intersection(set1, set2);
    ordered_array_set expectedSet =
ordered_array_set_create_from_array((int[]) {2, 3}, 2);
    assert(ordered_array_set_isEqual(result, expectedSet) == true);
    ordered_array_set_delete(set1);
    ordered_array_set_delete(set2);
    ordered_array_set_delete(result);
    ordered_array_set_delete(expectedSet);
}

void test_ordered_array_set_difference() {
    ordered_array_set set1 = ordered_array_set_create_from_array((int[]) {1,
2, 3, 4}, 4);
    ordered_array_set set2 = ordered_array_set_create_from_array((int[]) {2,
3, 5}, 3);
    ordered_array_set result = ordered_array_set_difference(set1, set2);
    ordered_array_set expectedSet =
ordered_array_set_create_from_array((int[]) {1, 4}, 2);
    assert(ordered_array_set_isEqual(result, expectedSet) == true);
    ordered_array_set_delete(set1);
    ordered_array_set_delete(set2);
    ordered_array_set_delete(result);
    ordered_array_set_delete(expectedSet);
}

void test_ordered_array_set_symmetricDifference() {
    ordered_array_set set1 = ordered_array_set_create_from_array((int[]) {1,
2, 3, 4}, 4);
    ordered_array_set set2 = ordered_array_set_create_from_array((int[]) {2,
3, 5}, 3);
    ordered_array_set result = ordered_array_set_symmetricDifference(set1,
set2);
    ordered_array_set expectedSet =
ordered_array_set_create_from_array((int[]) {1, 4, 5}, 3);
    assert(ordered_array_set_isEqual(result, expectedSet) == true);
    ordered_array_set_delete(set1);
    ordered_array_set_delete(set2);
    ordered_array_set_delete(result);
    ordered_array_set_delete(expectedSet);
}

void test_ordered_array_set_complement() {
    ordered_array_set set = ordered_array_set_create_from_array((int[]) {1,
2, 3, 4, 5}, 5);
    ordered_array_set universumSet =
ordered_array_set_create_from_array((int[]) {1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
10);
    ordered_array_set result = ordered_array_set_complement(set,
universumSet);
    ordered_array_set expectedSet =
ordered_array_set_create_from_array((int[]) {6, 7, 8, 9, 10}, 5);
    assert(ordered_array_set_isEqual(result, expectedSet) == true);
    ordered_array_set_delete(set);
    ordered_array_set_delete(universumSet);
    ordered_array_set_delete(result);
}

```

```
        ordered_array_set_delete(expectedSet);
    }

    void test() {
        test_ordered_array_set_in();
        test_ordered_array_set_isSubset();
        test_ordered_array_set_insert();
        test_ordered_array_set_union();
        test_ordered_array_set_deleteElement();
        test_ordered_array_set_intersection();
        test_ordered_array_set_difference();
        test_ordered_array_set_symmetricDifference();
        test_ordered_array_set_complement();
    }

    int main() {
        test();

        return 0;
    }
}
```

Задание №1. Определи маршрут (1056A)

Код задачи:

```
#include <stdio.h>
#include <assert.h>
#include <malloc.h>
#include <stdlib.h>

void append(int a[], size_t* n, int value) {
    a[*n] = value;
    (*n)++;
}

size_t linearSearch(const int a[], const size_t n, int x) {
    for (size_t i = 0; i < n; i++)
        if (a[i] == x)
            return i;
    return n;
}

typedef struct unordered_array_set {
    int* data;
    size_t size;
    size_t capacity;
} unordered_array_set;

unordered_array_set unordered_array_set_create(size_t capacity) {
    return (unordered_array_set) {malloc(sizeof(int) * capacity), 0,
    capacity};
}

void unordered_array_set_isAbleAppend(unordered_array_set *set) {
    assert(set -> size < set -> capacity);
}

size_t unordered_array_set_in(unordered_array_set* set, int value) {
    return linearSearch(set -> data, set -> size, value);
}

void unordered_array_set_insert(unordered_array_set* set, int value) {
    if (unordered_array_set_in(set, value) == set -> size) {
        unordered_array_set_isAbleAppend(set);
        append(set -> data, &set -> size, value);
    }
}

unordered_array_set unordered_array_set_intersection(unordered_array_set
set1, unordered_array_set set2) {
    size_t new_capacity = set1.size < set2.size ? set1.size : set2.size;
    unordered_array_set set = unordered_array_set_create(new_capacity);

    for (size_t i = 0; i < set1.size; i++)
        if (unordered_array_set_in(&set2, set1.data[i]) != set2.size)
            unordered_array_set_insert(&set, set1.data[i]);

    return set;
}
```

```

int main() {
    int n;
    scanf("%d", &n);

    int first_set_r;
    scanf("%d", &first_set_r);

    unordered_array_set set = unordered_array_set_create(first_set_r);
    for (int i = 0; i < first_set_r; i++) {
        int x;
        scanf("%d", &x);

        unordered_array_set_insert(&set, x);
    }

    for (int i = 0; i < n - 1; i++) {
        scanf("%d", &first_set_r);

        unordered_array_set subset = unordered_array_set_create(first_set_r);
        for (int j = 0; j < first_set_r; j++) {
            int x;
            scanf("%d", &x);

            unordered_array_set_insert(&subset, x);
        }


        set = unordered_array_set_intersection(set, subset);
    }

    for (int i = 0; i < set.size; i++) {
        printf("%d ", set.data[i]);
    }

    return 0;
}

```

Вердикт тестовой системы:

Основное									
№	Отправитель	Задача	Язык	Вердикт	Время	Память	Отослано	Протест.	
246377895	Дорешивание: Sasha39_-	1056A - 19	GNU C11	Полное решение	15 мс	468 КБ	2024-02-14 19:03:30	2024-02-14 19:03:30	 <div>Сравнить</div>

Задание №2. Пропущенная серия (440A)

Код задачи:

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>

typedef struct unordered_array_set {
    int* data;
    size_t size;
    size_t capacity;
} unordered_array_set;

unordered_array_set unordered_array_set_create(size_t capacity) {
    return (unordered_array_set) {malloc(sizeof(int) * capacity), 0,
capacity};
}

typedef struct ordered_array_set {
    int* data;
} ordered_array_set;

ordered_array_set ordered_array_set_create(size_t capacity) {
    return (ordered_array_set) {malloc(sizeof(int) * capacity), 0, capacity};
}

int main() {
    int n;
    scanf("%d", &n);

    ordered_array_set set = ordered_array_set_create(n);
    for (int i = 1; i < n + 1; i++) {
        set.data[i - 1] = i;
    }


    for (int i = 0; i < n - 1; i++) {
        int x;
        scanf("%d", &x);

        set.data[x - 1] = 0;
    }

    for (int i = 0; i < n; i++) {
        if (set.data[i] != 0) {
            printf("%d", i + 1);
        }
    }

    return 0;
}
```

Вердикт тестовой системы:

Основное									
№	Отправитель	Задача	Язык	Вердикт	Время	Память	Отослано	Протест.	
246452769	Дорешивание: Sasha39_-	440A - 9	GNU C11	Полное решение	31 мс	656 КБ	2024-02-15 11:45:26	2024-02-15 11:45:26	 <input type="button" value="Сравнить"/>

Задание №3. Перестановка букв (1093В)

Код задачи:

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>

typedef struct unordered_array_set {
    int* data;
    size_t size;
    size_t capacity;
} unordered_array_set;

unordered_array_set unordered_array_set_create(size_t capacity) {
    return (unordered_array_set) {malloc(sizeof(int) * capacity), 0,
capacity};
}

int main() {
    int t;
    scanf("%d", &t);

    int eng_alphabet = 26; //26 букв в англ алфавите

    for (int i = 0; i < t; i++) { //проход по множеству t
        unordered_array_set amount = unordered_array_set_create(26);
        for (int j = 0; j < eng_alphabet; j++) {
            amount.data[j] = 0;
            amount.size++;
        }

        char str[1000]; //по условию
        scanf("%s", str);


        size_t size_str = strlen(str);
        for (size_t i = 0; i < size_str; i++) {
            size_t char_index = str[i] - 'a';
            amount.data[char_index]++;
        }

        int is_palindrome = 1;
        for (size_t i = 0; i < eng_alphabet; i++) {
            if (amount.data[i] != 0 && amount.data[i] != size_str)
                is_palindrome = 0;
        }

        if (is_palindrome) {
            printf("-1\n");
        } else {
            for (int i = 0; i < eng_alphabet; i++)
                for (int j = 0; j < amount.data[i]; j++)
                    printf("%c", 'a' + i);
            printf("\n");
        }
    }

    return 0;
}
```

Вердикт тестовой системы:

Основное										
№	Отправитель	Задача	Язык	Вердикт	Время	Память	Отослано	Протест.		
246459230	Дорешивание: Sasha39-_-	1093B - 19	GNU C11	Полное решение	390 мс	408 КБ	2024-02-15 12:50:24	2024-02-15 12:50:24		<input type="button" value="Сравнить"/>

Задание №4. Тихий класс (1166A)

Код задачи:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct unordered_array_set {
    int* data;
    size_t size;
    size_t capacity;
} unordered_array_set;

unordered_array_set unordered_array_set_create(size_t capacity) {
    return (unordered_array_set) {malloc(sizeof(int) * capacity), 0,
capacity};
}

int count_couple(int n) {
    return (n * (n - 1)) / 2; // Это количество всех возможных пар числа n.
}

int main() {
    int eng_alphabet = 26; // 26 букв в англ алфавите

    unordered_array_set amount = unordered_array_set_create(eng_alphabet);
    for (size_t i = 0; i < eng_alphabet; i++) {
        amount.data[i] = 0;
        amount.size++;
    }

    int n;
    scanf("%d", &n);

    for (int n_sets = 0; n_sets < n; n_sets++) {
        char s[20]; // 20 букв по условию
        scanf("%s", s);

        size_t char_index = s[0] - 'a';
        amount.data[char_index]++;
    }


    int res = 0;
    for (size_t i = 0; i < eng_alphabet; i++) {
        int amount_first_class = amount.data[i] / 2;
        int amount_second_class = amount.data[i] % 2 == 0 ? amount.data[i] /
2 : amount.data[i] / 2 + 1;

        res += count_couple(amount_first_class) +
count_couple(amount_second_class);
    }

    printf("%d\n", res);

    return 0;
}
```

Вердикт тестовой системы:

Основное									
№	Отправитель	Задача	Язык	Вердикт	Время	Память	Отослано	Протест.	
246468412	Дорешивание: Sasha39_-	1166A - 33	GNU C11	Полное решение	15 мс	260 КБ	2024-02-15 14:27:06	2024-02-15 14:27:06	 <input type="button" value="Сравнить"/>

Задание №5. Щедрый Кефа (841A)

Код задачи:

```
#include <stdio.h>

int main() {
    int ammount_balls;
    int ammount_friends;
    scanf("%d %d", &ammount_balls, &ammount_friends);


    char balls[ammount_balls];
    scanf("%s", balls);

    int count_balls[123] = {0}; // от 0 до 122 - количество букв в таблице
    ASCII
    for(int i = 0; i < ammount_balls; i++){
        count_balls[balls[i]]++;
    }

    int flag = 1;
    for(int i = 'a'; i <= 'z'; i++) { // Перебрав в таблице ASCII - от 97 до
    122 включительно
        if(count_balls[i] > ammount_friends){
            flag = 0;
        }
    }
    if(flag){
        printf("YES\n");
    } else {
        printf("NO\n");
    }

    return 0;
}
```

Вердикт тестовой системы:

Основное									
№	Отправитель	Задача	Язык	Вердикт	Время	Память	Отослано	Протест.	
246505442	Дорешивание: Sasha39-_-	841A - 46	GNU C11	Полное решение	15 мс	264 КБ	2024-02-15 17:46:01	2024-02-15 17:53:54	 <input type="button" value="Сравнить"/>

Задание №6. Перекраска собачек (1025A)

Код задачи:

```
#include <stdio.h>
#include <malloc.h>

typedef struct ordered_array_set {
    int *data; // элементы множества
    size_t size; // количество элементов в множестве
    size_t capacity; // максимальное количество элементов в множестве,
    вместимость, мощность
} ordered_array_set;

ordered_array_set ordered_array_set_create(size_t capacity) {
    ordered_array_set set;
    set.data = (int *) malloc(capacity * sizeof(int));
    set.size = 0;
    set.capacity = capacity;
    return set;
}

int main() {
    int n;
    scanf("%d", &n);

    int en_alphabet = 26;

    ordered_array_set count_colors = ordered_array_set_create(en_alphabet);
    for (size_t i = 0; i < en_alphabet; i++) {
        count_colors.data[i] = 0;
    }

    char colors[n];
    scanf("%s", colors);

    for (int i = 0; i < n; i++) {
        count_colors.data[colors[i] - 'a']++;
    }

    int flag = 0;

    for(size_t i = 0; i < en_alphabet; i++) {
        if(n == 1 || count_colors.data[i] > 1) {
            flag = 1;
        }
    }

    if (flag)
        printf("YES\n");
    else
        printf("NO\n");

    return 0;
}
```

Вердикт тестовой системы:

Основное										
№	Отправитель	Задача	Язык	Вердикт	Время	Память	Отослано	Протест.		
246557841	Дорешивание: Sasha39-_-	1025A - 23	GNU C11	Полное решение	31 мс	344 КБ	2024-02-15 19:20:23	2024-02-15 19:20:23	★	<button>Сравнить</button>

Задание №7. Ступени (1011A)

Код задачи:

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#define eng_alphabet 26
typedef struct unordered_array_set {
    int* data;
    size_t size;
    size_t capacity;
} unordered_array_set;

unordered_array_set unordered_array_set_create(size_t capacity) {
    return (unordered_array_set) {malloc(sizeof(int) * capacity), 0,
capacity};
}

int get_minimum_total_weight_rocket(const int a[], int pos, int k) {
    int result = pos + 1;
    int amount_stages = 1;
    int amount_letter = eng_alphabet;
    int last = pos;

    for (int i = pos + 2; amount_stages < k && i < amount_letter; i++) {
        if (a[i] != 0 && i - last > 1) {
            result += i + 1;
            amount_stages++;
            last = i;
        }
    }

    if (amount_stages != k)
        return -1;
    else
        return result;
}

int main() {
    unordered_array_set amount = unordered_array_set_create(eng_alphabet);
    for (size_t i = 0; i < eng_alphabet; i++) {
        amount.data[i] = 0;
        amount.size++;
    }

    int ammount_available_teps, ammount_required_steps;
    scanf("%d %d", &ammount_available_teps, &ammount_required_steps);

    char str[ammount_available_teps];
    scanf("%s", str);

    for (size_t i = 0; i < ammount_available_teps; i++) {
        size_t char_index = str[i] - 'a';

        amount.data[char_index]++;
    }

    int index_first_element = 0;
    for (int i = 0; i < eng_alphabet; i++) {
        if (amount.data[i] != 0) {
            index_first_element = i;
            break;
        }
    }
}
```

```

    }

    int result = get_minimum_total_weight_rocket(amount.data,
index_first_element, ammount_required_steps);

    printf("%d", result);

    return 0;
}

```

Вердикт тестовой системы:

Основное										
№	Отправитель	Задача	Язык	Вердикт	Время	Память	Отослано	Протест.		
246575217	Дорешивание: Sasha39-_-	1011A - 28	GNU C11	Полное решение	30 мс	256 КБ	2024-02-15 20:55:37	2024-02-15 20:55:37	★	Сравнить

Задание №8. Башни (37A)

Код задачи:

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#define length_limit 1001//по условию не превосходит 1000 (но включая 1000)

typedef struct unordered_array_set {
    int* data;
    size_t size;
    size_t capacity;
} unordered_array_set;

unordered_array_set unordered_array_set_create(size_t capacity) {
    return (unordered_array_set) {malloc(sizeof(int) * capacity), 0,
capacity};
}

int getMax(const int a[], const size_t n) {
    int max = a[0];
    for (size_t i = 0; i < n; i++)
        if (a[i] > max)
            max = a[i];

    return max;
}

int main() {
    unordered_array_set amount = unordered_array_set_create(length_limit);
    for (size_t i = 0; i < length_limit; i++) {
        amount.data[i] = 0;
        amount.size++;
    }

    int ammountBars;
    scanf("%d", &ammountBars);

    for (size_t i = 0; i < ammountBars; i++) {
        int lengthBars;
        scanf("%d", &lengthBars);

        amount.data[lengthBars]++;
    }


    int max_tower_height_ammount = getMax(amount.data, amount.size);

    int total_amount_element = 0;
    for (size_t i = 0; i < length_limit; i++)
        if (amount.data[i] != 0)
            total_amount_element++;

    printf("%d %d\n", max_tower_height_ammount, total_amount_element);

    return 0;
}
```

Вердикт тестовой системы:

Основное										
№	Отправитель	Задача	Язык	Вердикт	Время	Память	Отослано	Протест.		
246577834	Дорешивание: Sasha39-_-	37A - 10	GNU C11	Полное решение	62 мс	268 КБ	2024-02-15 21:15:57	2024-02-15 21:15:57		Сравнить

Задание №9. Бейджик (1020В)

Код задачи:

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>

typedef struct unordered_array_set {
    int* data;
    size_t size;
    size_t capacity;
} unordered_array_set;

unordered_array_set unordered_array_set_create(size_t capacity) {
    return (unordered_array_set) {malloc(sizeof(int) * capacity), 0,
capacity};
}

int getReference(const int a[], const int n, int pos) {
    int *visited = malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        visited[i] = 0;
    }
    int current = pos;
    visited[current] = 1;

    while (!visited[a[current]]) {
        visited[a[current]] = 1;
        current = a[current];
    }

    int result = a[current] + 1;

    free(visited);

    return result;
}

int main() {
    int ammount_pupils;
    scanf("%d", &ammount_pupils);


    int pupils[ammount_pupils];
    for (size_t i = 0; i < ammount_pupils; i++) {
        int x;
        scanf("%d", &x);

        pupils[i] = x - 1;
    }

    for (int pos = 0; pos < ammount_pupils; pos++) {
        printf("%d ", getReference(pupils, ammount_pupils, pos));
    }

    return 0;
}
```

Вердикт тестовой системы:

Основное									
№	Отправитель	Задача	Язык	Вердикт	Время	Память	Отослано	Протест.	
246581263	Дорешивание: Sasha39_-	1020В - 16	GNU C11	Полное решение	31 мс	260 КБ	2024-02-15 21:45:47	2024-02-15 21:45:47	 <input type="button" value="Сравнить"/>

Задание №10. Разнообразие – это хорошо (672В)

Код задачи:

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#define eng_alphabet 26

typedef struct unordered_array_set {
    int* data;
    size_t size;
    size_t capacity;
} unordered_array_set;

unordered_array_set unordered_array_set_create(size_t capacity) {
    return (unordered_array_set) {malloc(sizeof(int) * capacity), 0,
capacity};
}

int main() {
    unordered_array_set amount = unordered_array_set_create(eng_alphabet);
    for (size_t i = 0; i < eng_alphabet; i++) {
        amount.data[i] = 0;
        amount.size++;
    }

    int line_length;
    scanf("%d", &line_length);

    char str[line_length];
    scanf("%s", str);

    for (size_t i = 0; i < line_length; i++) {
        size_t char_index = str[i] - 'a';

        amount.data[char_index]++;
    }

    int res = 0;
    for (size_t i = 0; i < eng_alphabet; i++) {
        if (amount.data[i] > 1)
            res += amount.data[i] - 1;
    }

    if (line_length > eng_alphabet)
        printf("-1\n");
    else
        printf("%d", res);

    return 0;
}
```

Вердикт тестовой системы:

Основное										
№	Отправитель	Задача	Язык	Вердикт	Время	Память	Отослано	Протест.		
246582698	Дорешивание: Sasha39_-	672В - 41	GNU C11	Полное решение	31 мс	344 КБ	2024-02-15 21:59:37	2024-02-15 21:59:37	★	<button>Сравнить</button>

Задание №11. Игра: Банковские карты (777B)

Код задачи:

```
#include <stdlib.h>
#include <stdio.h>

typedef struct unordered_array_set {
    int *data;
    size_t size;
    size_t capacity;
} unordered_array_set;

//компаратор
int compareInts(const void *a, const void *b) {
    int arg1 = *(const int *) a;
    int arg2 = *(const int *) b;
    if (arg1 < arg2) return -1;
    if (arg1 > arg2) return 1;
    return 0;
}

unordered_array_set unordered_array_set_create(size_t capacity) {
    unordered_array_set set;
    set.data = (int *)malloc(capacity * sizeof(int));
    set.size = capacity;
    set.capacity = capacity;
    return set;
}

unordered_array_set unordered_array_set_create_from_array(char *a, size_t n)
{
    unordered_array_set set = unordered_array_set_create(n);
    for (int i = 0; i < n; i++) {
        set.data[i] = a[i] - '0';
    }
    qsort(set.data, set.size, sizeof(int), compareInts);

    return set;
}

void unordered_array_set_delete(unordered_array_set set) {
    free(set.data);
}

int main() {
    int ammount_digits;
    scanf("%d", &ammount_digits);

    char Sherlock[ammount_digits];
    char Moriarty[ammount_digits];

    scanf("%s", Sherlock);
    scanf("%s", Moriarty);

    unordered_array_set set_S =
    unordered_array_set_create_from_array(Sherlock, ammount_digits);
    unordered_array_set set_M =
    unordered_array_set_create_from_array(Moriarty, ammount_digits);
```

```

int min_ammount_flick = 0;
int max_ammount_flick = 0;

int j = 0;

for (int i = 0; i < ammount_digits; i++) {
    while (j < ammount_digits && set_M.data[j] < set_S.data[i])
        j++;

    if (j < ammount_digits)
        j++;
    else
        min_ammount_flick++;
}

j = 0;

for (int i = 0; i < ammount_digits; i++) {
    while (j < ammount_digits && set_M.data[j] <= set_S.data[i])
        j++;

    if (j < ammount_digits) {
        max_ammount_flick++;
        j++;
    }
}


printf("%d\n", min_ammount_flick);
printf("%d\n", max_ammount_flick);

unordered_array_set_delete(set_M);
unordered_array_set_delete(set_S);

return 0;
}

```

Вердикт тестовой системы:

Основное									
№	Отправитель	Задача	Язык	Вердикт	Время	Память	Отослано	Протест.	
246585486	Допешивание: Sasha39-_-	777B - 32	GNU C11	Полное решение	15 мс	268 КБ	2024-02-15 22:27:21	2024-02-15 22:27:21	 <div>Сравнить</div>

Задание №12. Противоположности притягиваются (131B)

Код задачи:

```
#include <stdio.h>
#include <stdlib.h>
#define T_client_par 22 //от -10 до 10
typedef struct unordered_array_set {
    long long *data;
    size_t size;
    size_t capacity;
} unordered_array_set;

unordered_array_set unordered_array_set_create(size_t capacity) {
    unordered_array_set set;
    set.data = (long long *)malloc(capacity * sizeof(long long));
    set.size = capacity;
    for (size_t i = 0; i < set.size; i++)
        set.data[i] = 0;

    return set;
}

int main() {
    int ammount_clients;
    scanf("%d", &ammount_clients);

    unordered_array_set array_set = unordered_array_set_create(T_client_par);

    long long sum;

    for (int i = 0; i < ammount_clients; i++) {
        int client_parameter;
        scanf("%d", &client_parameter);

        array_set.data[client_parameter + 10]++;
    }


    sum = array_set.data[10] * (array_set.data[10] - 1) / 2;

    for (int i = 0; i < 10; i++)
        sum = sum + array_set.data[i] * array_set.data[20 - i];

    printf("%lld", sum);

    return 0;
}
```

Вердикт тестовой системы:

Основное									
№	Отправитель	Задача	Язык	Вердикт	Время	Память	Отослано	Протест.	
246587718	Дорешивание: Sasha39_-	131B - 31	GNU C11	Полное решение	62 мс	260 КБ	2024-02-15 22:51:52	2024-02-15 22:51:52	 <input type="button" value="Сравнить"/>

Вывод: в ходе выполнения лабораторной работы я закрепил навыки работы со структурами, изучил «простые» способы представления множеств в памяти ЭВМ и создал библиотеки с полезными функциями.