Задача A. Set (!) (2 балла)

Пример

set.in	set.out
insert 2 insert 5 insert 3 exists 2 exists 4 insert 2 delete 2 exists 2	true false false

В этой задаче я буду использовать структуру данных хеш-таблицы (hash table).

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

Этот метод ускорит поиск O (1) в оптимальном случае и O (N), если все элементы находятся в одном единственном связанном списке (single linked list).

```
struct node *hTable[Size];

void initbuckets(){
    for (int i=0; i<Size; i++) {
        hTable[i] = NULL;
    }
}
```

-> Я сгенерирую hashtable и инициализирую все его значения NULL значением

```
struct node* newNode(int x){
    struct node* temp=(struct node*)malloc(sizeof(struct node));
    temp->val = x;
    temp->next = NULL;
    return temp;
}
```

-> Эта функция используется для создания нового узла

```
bool exists (int x){
    int key = abs(x%Size);
    for (struct node* p = hTable[key]; p!=NULL; p=p->next) {
        if (p->val == x) {
            return true;
        }
    }return false;
}
```

-> это функция для поиска числа Мы рассмотрим каждый элемент single linked list в bucket key

```
avoid insert(int x){
    if (exists(x)) {
        return;
    }else{
        int key = abs(x%Size);
        if (hTable[key] == NULL) {
            hTable[key] = newNode(x);
            return;
        }else{
            struct node* p =
newNode(x);
            p->next = hTable[key];
            hTable[key] = p;
            return;
        }
    }
```

-> Это функция, используемая для добавления числа в hashtable

- Если x найден в хеш-таблице, ничего не делать

- Наоборот

+ Если bucket k в hashtable пуст, то мы создаем новый узел из x и добавляем в bucket

+ в противном случае мы добавим его в начало односвязного списка (single linked list)

```
void deleteNode(int x){
    if (!exists(x)) {
        return;
    }else{
        int key = abs(x%Size);
        if (hTable[key]->val == x) {
            hTable[key] = hTable[key]-
>next;
        }else{
            hTable[key]->next = NULL;
        }
    }
```

-> Это функция для удаления

- Если x не может быть найден, ничего не делать

- в противном случае я бы удалил его из односвязного списка