Задача 1. Множество целых чисел

Источник:	базовая I
Имя входного файла:	`---`
Имя выходного файла:	`---`
Ограничение по времени:	3 секунды*
Ограничение по памяти:	разумное

Требуется написать структуру данных для хранения множества целых чисел, которая позволяет быстро узнать, лежит ли заданное число в множестве. Структуру данных надо оформить в виде динамической библиотеки.

В систему проверки требуется отправить:
- Скрипт сборки `c.sh.txt`.
- Хедер `integerset.h`.
- Любые дополнительные файлы хедеров и исходного кода.

Скрипт должен собирать динамическую библиотеку `integerset.so`, с которой будет собираться тестирующий код жюри.

В хедере `integerset.h` должен быть объявлен тип `IntegerSet`, а также две функции `CreateSet` и `IsInSet`. Сигнатуру этих функций и принципы работы следует определить самостоятельно, исходя из примера использования, приведённого на следующей странице.

Гарантируется, что размер создаваемого множества всегда будет неотрицательным. Также гарантируется, что в `IsInSet` будет передаваться либо указатель, который ранее вернула функция `CreateSet`, либо ноль. Количество вызовов каждой функции не будет превышать $3 \cdot 10^5$. Размер каждого множества не будет превышать 10^5. Сумма размеров всех созданных множеств не будет превышать $3 \cdot 10^5$. Освобождать память из под множеств в этой задаче **не** нужно.

Пример кода, который должен собираться вместе с библиотекой импорта и работать:

```
#include "integerset.h"
#include <assert.h>
#include <limits.h>

int main() {
    int myarr[] = {1, 2, 3};
    //create set with numbers 1, 2, and 3
    IntegerSet *s123 = CreateSet(myarr, 3);
    myarr[1] = -5;
    assert(IsInSet(s123, 3) == 1 && IsInSet(s123, 2) == 1);
    assert(IsInSet(s123, 0) == 0 && IsInSet(s123, 4) == 0);
    //create set with numbers -5 and 3
    IntegerSet *s15 = CreateSet(myarr + 1, 2);
    assert(IsInSet(s15, 3) == 1 && IsInSet(s15, -5) == 1);
    assert(IsInSet(s15, 1) == 0 && IsInSet(s15, 2) == 0);
    //create empty set (note: null pointer is allowed only when size is 0)
    IntegerSet *sEmpty = CreateSet(0, 0);
    assert(sEmpty != 0);
    assert(IsInSet(sEmpty, 0) == 0 && IsInSet(sEmpty, -123456789) == 0);
    IntegerSet *sNull = 0;
    //null pointer must be treated by IsInSet as empty set
    assert(IsInSet(sNull, 0) == 0 && IsInSet(sNull, -123456789) == 0);
    myarr[0] = -5;
    //if array has equal elements, then CreateSet must return 0
    IntegerSet *sBad = CreateSet(myarr, 3);
    assert(sBad == 0);
    int largearr[] = {1, 5, INT_MAX, INT_MIN, 1000000000, -123, 555, 0};
    //create set with 8 numbers from largearr
    IntegerSet *sLarge = CreateSet(largearr, 8);
    assert(IsInSet(sLarge, INT_MAX) == 1 && IsInSet(sLarge, INT_MIN) == 1);
    assert(IsInSet(sLarge, 1000000000) == 1 && IsInSet(sLarge, -123) == 1);
    assert(IsInSet(sLarge, 123) == 0 && IsInSet(sLarge, -5) == 0);
    largearr[1] = 5;
    sLarge = CreateSet(largearr, 8); //same set as previously
    assert(IsInSet(sLarge, 5) == 1);
    largearr[7] = 5;
    sLarge = CreateSet(largearr, 8); //now it has equal elements
    assert(sLarge == 0);
    return 0;
}
```