§2 Второй отборочный этап

2.1 Задачи по физике

Задача 2.1.1

Условие:

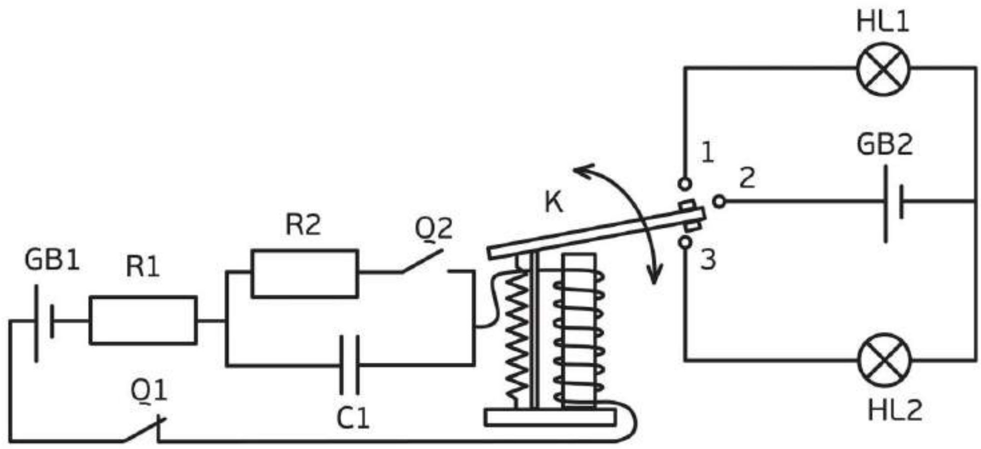


Рис. 1

Схема:

На рисунке 1 изображена электрическая цепь, включающая в себя: два резистора R_1 и R_2, плоский конденсатор C_1, электромагнитное реле K, два источника питания GB_1 и GB_2, две сигнализационные лампы HL_1 и HL_2, связанные между собой выключатели Q_1 и Q_2.

О схеме необходимо знать следующее:

1. При замыкании Q_1 размыкается Q_2, и наоборот (время коммутации считать равным нулю).
2. При замыкании Q_1 первоначально замыкаются контакты 2 и 3. При замыкании Q_2 первоначально замыкаются контакты 1 и 2.
3. В начальный момент каждого опыта Q_1 замыкнут.
4. Полный заряд конденсатора принять как 99,2% емкости.

Известные величины:

R_1 = 1 кОм;
R_2 = 1,4 кОм;
$t_{ср.\ реле}$ = 20 мс - время срабатывания реле;
$t_{отп.\ реле}$ = 10 мс - время отпускания реле;
$\varepsilon_0 = 8{,}85 \cdot 10^{-12}$ ф/м;
Площадь пластин конденсатора $S = 2 \cdot 10^{-2}$ м^2;
Расстояние между пластинами $d = 17{,}7 \cdot 10^{-9}$ м;
Диэлектрик конденсатора – воздух ($\varepsilon = 1$).

Требуется определить:

1. Каким нужно выбрать R_2, чтобы конденсатор полностью разряжался за время отпуска реле? Ответ дать в омах.
2. Как долго будет гореть HL_2 в ходе эксперимента, до момента самопроизвольного переключения органа реле K с контактов 2 и 3 на контакты 1 и 2? Ответ дать в секундах.

3. На сколько процентов будет заряжен конденсатор по прошествии 0,172 мс. с начала эксперимента при условии, что его необходимо разряжать и заряжать полностью. При необходимости округлить до десятых.

Ход решения:

1. Исходя из того, что полный заряд конденсатора 99,2% есть 5*T, где T= C*R принимаем малый контур за контур, в котором будет проходить разрядка конденсатора. 5*T = 5*C*R_2. Из условия принимаем время полного разряда конденсатора за время отпуска реле - 5*T = 0,01с. Имеем 0,01 = 5*C*R_2, следовательно R2 = $\frac{0{,}01}{5*C}$ = **200 Ом.**

2. Полный заряд конденсатора 99,2% есть 5*T, где T= C*R.

Учитывая контур заряда используем R_1, получаем полное время заряда 5*C*R_1 = 5*($\frac{\varepsilon_0 * \varepsilon * S}{d}$)*$10^3$ = 5*($\frac{8{,}85*10^{-12} * 1*2*10^{-2}}{17{,}7*10^{-9}}$)*10^3 = 0,05 с. Исходя из условия того, что ток в цепи будет протекать до момента зарядки конденсатора имеем, что ток в цепи будет протекать 0,05 сек. Однако, при включении схемы и протекании тока по реле часть времени расходуется на процесс срабатывания ($t_{ср.\ реле}$ = 20 мс), учитывая это получаем, что время горения HL_2 до момента самопроизвольного переключения органа реле K с контактов 2 и 3 на контакты 1 и 2 5*C*R_1 - $t_{ср.\ реле}$ = 0,05 - 0,02 = **0,03 с.**

3. Вкратце:

- Q_1 включается в начале эксперимента, HL_2 горит время, вычисленное ранее – 0,03 с.

- После этого включается Q_2, конденсатор разряжается за время 5*C*R_2 = 5*10^{-5}*1,4*10^3 = 7*10^{-2}с. = 0,07 с. Это происходит медленней, чем время отпуска реле (0,01с), в соответствии с чем учитываем только время 0,07.

- Снова включается Q_1, HL_2 горит время, вычисленное ранее – 0,03 с.

прошло 0,13 с. (130 мс.)

осталось 0,042 с. = 42 мс. на разряд конденсатора

3*T=3*C*R_2 = 3*10^{-5}*1,4*10^3= 4,2*10^{-2}с. = 0,042 с. = 42 мс. (разрядка до 95%)

0,13 + 0,042 с. = 0,172с. = 172 мс.

Имеем, что этого времени хватит, чтобы разрядить конденсатор на 95%.

Получается в нем остается 99,2% - 95% = **4,2 %**

Ответ: 4,2 %

Задача 2.1.2

Условие:

Имеется пространство 21 на 21 клетку, размер каждой клетки которого 1 на 1 метр. На оси X установлены источники лучей A и B так, что направлены к центру пространства при условии углов между лучами и осью X α 1 =α 2 =45 0 . Исходная среда имеет показатель преломления n=1. Из нее луч A последовательно попадает в среду 1, затем в среду 3, в то время как луч B попадает в среду 2, затем также в среду 3. В среде 3 помещена рассеивающая линза с главным фокусным расстоянием F = 1,2м.

Найдите координаты точки C - пересечения преломленных лучей A и B,

округлите до целых чисел. Показатели преломления сред: n 1 =1,582; n 2 =2,909; n 3 =0,850. Координаты источников лучей: (1;0) A и (20;0) B (т.е. на оси X) Координаты сред: n - (0-21;0- 4); n 1 - (0-11;4- 8); n 2 - (11-21;4- 8); n 3 - (0-21;8- 21);

Координаты местонахождения линзы: (1-20; 10) (т.е. параллельно оси Y).

Считать, что точка C находится на главной оптической оси линзы.

Решение:

1. В соответствии с законом преломления имеем: $\frac{sin\alpha_1}{sin\beta_1} = \frac{n_1}{1}; \frac{sin\alpha_2}{sin\beta_2} = \frac{n_2}{1}; \frac{sin\beta_1}{sin\gamma_1} = \frac{n_3}{n_1}; \frac{sin\beta_2}{sin\gamma_2} = \frac{n_3}{n_2}$.

Исходя из условия α_1=α_2=45^0, $sin\beta_1 = \frac{sin\alpha_1}{n_1} = \frac{sin45^0}{1,582} = \frac{0,707}{1,582} = 0,447$, $sin\beta_2 = \frac{sin\alpha_2}{n_2} = \frac{sin45^0}{2,909} = \frac{0,707}{2,909} = 0,243$. $sin\gamma_1 = \frac{sin\beta_1 * n_1}{n_3} = \frac{0,447*1,582}{0,850} = 0,832$, $sin\gamma_2 = \frac{sin\beta_2 * n_2}{n_3} = \frac{0,243*2,909}{0,850} = 0,832$. Получаем координаты попадания лучей на линзу.

2. Так как линза рассеивающая, имеем $\frac{1}{F} = \frac{1}{d} - \frac{1}{f}$. из построенного рисунка видим, что d = 1. $\frac{1}{1,2} = \frac{1}{1} - \frac{1}{f}$. Тогда f = 6м.

Ответ:

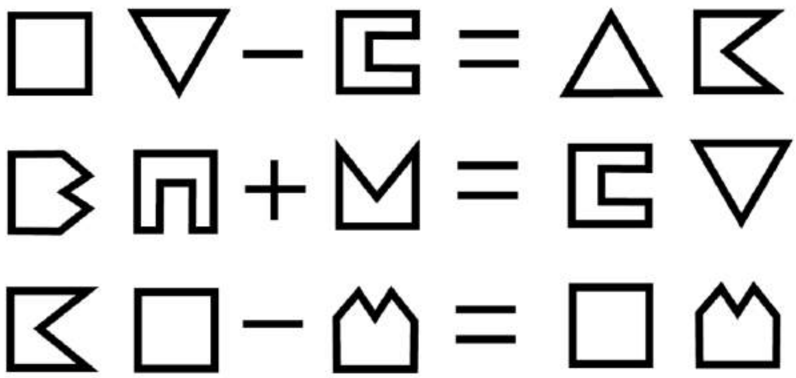
11 по оси X ,16 по оси Y, или просто **11,16.**

Задача 2.1.3

Условие:

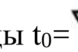
Очень маленький объект находится в льдинке, в центре теплоизолированной емкости, наполненного водой. Сам объект соединен с дном емкости посредством легкой нитки. При условии того, что тепловое равновесие устанавливается мгновенно, найдите температуру воды в тот момент, когда сила натяжения нити станет нулевой.

Часть данных задачи оказалась зашифрованной, но имеется система уравнений, способная доопределить неизвестные. Элементы являются числами от 0 до 9.



Дано:

Масса льдинки M_0 = 100г.

Масса воды m_0 = 0 г.

Температура воды t_0=0^0C

Удельная теплоемкость воды с = 4200 Дж/(кг*^{0}C)

Температура льда и объекта t_1=0^0C

Сила натяжения нити в начальный момент T = 0,0Н

Плотность воды ρ = 1000 кг/м^3

Плотность льда ρ_1 = 900 кг/м^3

Плотность объекта ρ_2 = 200 кг/м^3

Удельная теплота плавления льда λ = кДж/кг.

Решение:

Каждый символ обозначает лишь число, равное кол-ву углов символа.

Масса льдинки M_0 = 100г.

Масса воды m_0 = 750 г.

Температура воды $t_0 = 30^0C$

Удельная теплоемкость воды с = 4200 Дж/(кг*^{0}С)

Температура льда и шарика $t_1 = 0^0C$

Сила натяжения нити в начальный момент T = 0,08

Плотность воды ρ = 1000 кг/м^3

Плотность льда ρ_1 = 900 кг/м^3

Плотность алюминия ρ_2 = 2700 кг/м^3

Удельная теплота плавления льда λ = 335 кДж/кг

Ход решения:

Сила натяжения нити станет равной нулю, когда часть льда растает и уменьшится выталкивающая сила. Из условия равновесия системы в исходном состоянии находим массу *m* объекта:

$T + (M_0 + m) - F_A = 0$,

$F_A = (V_1 + V_2)\rho g$,

$V_1 = \frac{M_0}{\rho_1}$, $V_2 = \frac{\square}{\rho_2}$,

$T + (M_0 + m)\, g - (\frac{M_0}{\rho_1} + \frac{m}{\rho_2})\, \rho g = 0$,

$mg(1 - \frac{\square}{\rho_2}) = M_0 g(\frac{\square}{\rho_1} - 1) - T$,

$m = \frac{M_0(\frac{\rho}{\rho_1} - 1) - \frac{T}{g}}{(1 - \frac{\rho}{\rho_2})} = 0{,}0049$ кг.

Сила натяжения нити $T = (\frac{M_0}{\rho_1} + \frac{m}{\rho_2})\, \rho g - (M_0 + m)\, g$ обратится в ноль, если масса льда уменьшится до некоторого значения M_1, удовлетворяющего условию: $M_0 + m = \rho(\frac{M_1}{\rho_1} + \frac{m}{\rho_2})$, откуда $M_1 = \frac{m(1 - \frac{\square}{\rho_2})}{\frac{\rho}{\rho_1} - 1} = \frac{m(1 - \frac{1000}{2700})}{\frac{1000}{900} - 1} =$ 0,028 кг.

Значит, для исчезновения натяжения силы натяжения должно быть расплавлено $\Delta M = M_0 - M_1$ = 0,100-0,028 = 0,072 кг льда. Так как он уже находится при температуре плавления для этого необходимо $Q_1 = \Delta M\lambda = 0{,}072 * 335 * 10^3 =$ 24120Дж. Эта энергия будет получена за счет охлаждения воды. В итоге в системе установится тепловое равновесие при температуре t_2, определяемой из уравнения теплового баланса:

$cm_0(t_0 - t_2) = Q_1 + c(M_0 - M_1)(t_2 - 0^0C)$

Отсюда находим $t_2 = \frac{cm_0t_0 - Q_1}{c[m_0 + \Delta M]} = \frac{4200 * 0{,}75 * 30 - 24120}{4200[0{,}75 + 0{,}072]} = \frac{70380}{3452{,}4} = 20{,}39\ ^0C$.

Ответ: $20{,}39^0C$

2.2 Задачи по информатике

Задача 2.2.1 (5 баллов)

Условие:

Давным-давно, когда Даша была ребёнком, а её дом ещё не был умным, в стену её комнаты был встроен сейф с кодовым замком. Папа Даши хотел, чтобы девочка выросла и стала математиком, поэтому часто загадывал ей математические загадки. Ответом, обычно, была последовательность чисел. Эту последовательность папа устанавливал в качестве кода от замка сейфа, клал в него новые игрушки, а решив задачку, Даша могла их достать.

Однажды папа загадал Даше такую задачку, которую она так и не смогла решить. Сейф так и остался закрытым.

Недавно Даша приехала в гости к родителям, и вспомнила об этом сейфе и той загадке. Её условия были следующие. Для заданного n найти целые положительные числа a, b, c, являющиеся решением уравнения:

$a^n + b^n - c^n = 0.$

Даше стало интересно, какой же сюрприз в детстве спрятал её отец. Спустя столько лет, Даше не составило труда найти решение и открыть сейф. А сможете ли вы решить эту задачку?

Формат входных данных

В единственной строке дано число n $(0 \leq n \leq 100)$.

Формат выходных данных

В единственной строке вывести три целых числа a, b и c. Если решений несколько, вывести такое, где величина a минимальна. Если и таких решений несколько, вывести такое, где величина b минимальна. Если решения не существует, вывести “-1” (без кавычек).

Пример 1

Входные данные	Выходные данные
0	-1

Пример 2

Входные данные	Выходные данные
1	1 2 3

Решение:

Для решения задачи воспользуемся Великой теоремой Ферма, которая утверждает, что для любого натурального числа $b > 2$ уравнение $a^{n} + b^n = c^n$ не имеет решения в целых ненулевых числах a, b, c. Таким образом, в качестве ответа при $n = 1$ необходимо вывести "$1\ 2\ 3$", при $n = 2$ - "$3\ 4\ 5$", а для всех остальных значений n вывести "-1".

Код решения (C++)

```
#include <iostream>
using namespace std;

int main()
{
    int n;
    cin >> n;
    if (n == 1) {
        cout << "1 2 3" << endl;
    }
    else if (n == 2) {
        cout << "3 4 5" << endl;
    }
    else {
        cout << -1 << endl;
    }

    return 0;
}
```

Задача 2.2.2 (15 баллов)

Условие:

Егор любит всё упорядочивать. Но ещё больше он любит придумывать критерии для упорядочивания. Например, он придумал несколько способов расставить книги на полке: в алфавитном порядке, в порядке возрастания размера книги, в порядке возрастания массы книги.

Но по таким скучным критериям упорядочивать книги Егору надоело. Тогда он придумал новый способ. Сначала Егор даёт каждой из N книг порядковый номер от 1 до N. Затем для каждой книги вводит число p_i такое, что $|p_i|$ – число страниц в книге -ой книге. Сами p_i могут быть отрицательными. Каждой книге Егор ставит в соответствие число $X_i = \Sigma_j |p_i - p_j|$ (т.е. сумму разностей количества страниц (с учётом знака при p_i) -ой книги со всеми остальными книгами). После этого все книги Егор выставляет именно в порядке возрастания величины X_i. Если у нескольких книг величина X совпадает, первой берётся та книга, порядковый номер которой меньше.

Егору быстро надоело выставлять книги в обозначенном порядке, потому что ему в голову пришла мысль о новом способе расстановки тарелок в сушилке. Дадим Егору поразвлечься с тарелками, а сами расставим книги так, как задумывал Егор!

Формат входных данных

В первой строке дано целое число N – количество книг ($0 \leq N \leq 10^5$).

В следующей строке содержится N целых чисел p_i – количество страниц в каждой из книг по порядку ($|p_i| \leq 10^6$).

Формат выходных данных

В единственной строке выведите N чисел – номера книг в том порядке, в котором они должны быть упорядочены.

Пример

Входные данные	Выходные данные
5 0 1 2 3 4	2 1 3 0 4

Условия

Ограничение по памяти: 256 МБ

Ограничение по времени: 1 с

Решение:

По условиям задачи требуется отсортировать книги в порядке возрастания суммы разностей $X_{\square}$ величины $p_{\square}$. При заданных ограничениях необходимо использовать любой метод сортировки со сложностью не превышающей $O(N \cdot logN)$.

Величины сумм разностей $X_{\square}$ заранее неизвестны. Простой способ посчитать их – для каждой из N книг перебрать все остальные $N - 1$ книги, суммируя разности по указанной формуле. Однако такой подход требует $O(N^2)$ операций, что приводит к слишком долгой работе алгоритма, превышающей ограничение по времени.

Для быстрого подсчёта воспользуемся следующим подходом. Предварительно отсортируем все элементы в порядке возрастания величины $p_{\square}$. Тогда для каждого i-го элемента (кроме первого и последнего) справедливо выражение:

$$X_{\square} = (X_{i-1} + (i-1) \cdot (p_i - p_{i-1})) + (X_{i+1} + (N-i) \cdot (p_{i+1} - p_i)), i = 2..(N-1).$$

Для первого и последнего элементов часть слагаемых опускается:

$$X_1 = (X_2 + (n-i) \cdot (p_2 - p_1)),$$

$$X_{\square} = (X_{N-1} + (i-1) \cdot (p_N - p_{N-1})).$$

Совокупность формул, представленных выше, показывает рекуррентную зависимость величины $X_{\square}$ от величин X_{i-1} и X_{i+1}. Для их подсчёта линейно пройдёмся по всем элементам в двух направлениях (по возрастанию i от 2 до N, и по убыванию от $N - 1$ до 1), при каждом проходе считая только сумму в соответствующем направлении. После чего третьим проходом просуммируем обе суммы по направлениям. Таким образом, все величины $X_{\square}$ будут посчитаны за $O(N)$. Учитывая, что такой подход требует предварительной сортировки, суммарная сложность поиска $X_{\square}$ есть $O(N \cdot logN)$, как и конечная сложность работы алгоритма.

Код решения (C++)

```
#define _CRT_SECURE_NO_WARNINGS

#include <iostream>
#include <cstdio>
#include <string>
```

```
#include <vector>
#include <utility>
#include <algorithm>
#include <ctime>

using namespace std;

typedef struct Auxx {
    int id;
    long long val;
    long long sum_back;
    long long sum_front;

    Auxx()
        : id(0)
        , val(0)
        , sum_back(0)
        , sum_front(0) {}

    Auxx(int _id, long long _val)
        : id(_id)
        , val(_val)
        , sum_back(0)
        , sum_front(0) {}

} Auxx;

bool less1(const Auxx &a, const Auxx &b)
{
    return a.val < b.val || (a.val == b.val && a.id < b.id);
}

bool less2(const Auxx &a, const Auxx &b)
{
    if (a.sum_back + a.sum_front < b.sum_back + b.sum_front)
        return true;

    if (a.sum_back + a.sum_front == b.sum_back + b.sum_front && a.id <
b.id)
        return true;

    return false;
}

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

    vector <Auxx> v(n);

    for (int i = 0; i < n; i++)
    {
        int tmp;
        scanf("%d", &tmp);
        v[i] = Auxx(i + 1, tmp);
```

```
        }

        sort(v.begin(), v.end(), less1);

        for (int i = 1; i < n; i++)
        {
            v[i].sum_back = v[i - 1].sum_back + i * (v[i].val - v[i - 1].val);
            v[n - i - 1].sum_front = v[n - i].sum_front + i * (v[n - i].val -
v[n - i - 1].val);
        }

        sort(v.begin(), v.end(), less2);

        for (int i = 0; i < n; i++)
            printf("%lld ", v[i].id);

        return 0;
    }
```

Задача 2.2.3 (20 баллов)

Условие:

Как-то раз Костя и его младшая сестра Света гуляли по каменистому пляжу реки. Света обратила внимание на лежащий у неё под левой ногой очень красивый прозрачный красный с буроватым оттенком камень. Костя сказал, что это был рубин. Девочке так сильно понравился этот камень и это приятное уху название, что она начала расспрашивать своего брата о том, какие ещё красивые камни бывают в природе. Но к великому её сожалению, познания Кости на этом почти и заканчивались.

Света не расстроилась, а наоборот, заинтересовалась темой драгоценных камней и в будущем очень сильно увлеклась геммологией. Но что ещё более важно, в тот день Света решила начать коллекционировать камни. Она спросила Костю, можно ли ей взять несколько камней с пляжа домой, на что Костя ответил, что не стоит Свете заниматься ерундой. Тогда девочка сильно расстроилась, а её брат предложил ей сыграть в игру с условием, что если она выиграет, он разрешит ей забрать несколько камней с собой.

Костя выложил в ряд N кучек камней, в -ой из которых было Q_i камней. В любой момент времени Костя мог выполнить одно из двух действий:

1) Добавить или убрать из какой-либо кучки некоторое количество камней.
2) Выбрать две кучки, и попросить Свету назвать количество камней во всех кучках, лежащих между двумя, выбранными Костей (включая сами выбранные кучки).

Света выигрывала в том случае, если после M действий Кости она ни разу не ошиблась с количеством камней. В тот день ей удалось выиграть. А вам предстоит написать программу, которая будет за вас играть в эту увлекательнейшую игру.

Формат входных данных

В первой строке дано два целых числа N и M – количество кучек и количество действий Кости соответственно ($1 \leq N, M \leq 10^5$).

Во второй строке дано N целых неотрицательных чисел, не превосходящих 10 – количество камней в каждой кучке.

В следующих M строках описаны действия Кости.

Действие изменения кучки описывается строкой, начинающейся с символа “M”, за которым через пробел следуют два числа: номер изменяемой кучки, и новое количество камней в этой кучке. То есть строка вида “M 5 9” означает, что в кучке с номером 5 теперь будет 9 камней. Костя никогда не добавлял слишком много камней, и в каждой кучке не могло оказаться больше 10 камней.

Действие вопроса к Свете описывается строкой, начинающейся с символа “S”, за которым через пробел следуют два числа: номера кучек, между которыми нужно посчитать камни. То есть строка вида “S 2 4” означает, что необходимо посчитать, сколько всего камней в кучках с номерами 2, 3 и 4.

Формат выходных данных

Для каждого действия-вопроса в отдельной строке выведите суммарное количество камней между заданными кучками.

Пример

Входные данные	Выходные данные
9 10 1 1 2 3 5 0 4 9 4 S 2 4 S 8 8 S 0 8 S 4 5 M 5 9 S 0 8 S 4 5 M 0 7 S 1 8 S 0 5	10 4 29 5 38 14 37 27

Условия

Ограничение по памяти: 256 МБ

Ограничение по времени: 1 с

Решение:

Для решения задачи воспользуемся структурой данных, называемой деревом отрезков. Дерево является бинарным. Каждый узел соответствует некоторому подотрезку [a, b] всего диапазона значений [0, N-1]. Корню дерева соответствует отрезок, равный всему исходному массиву, т.е. [0, N-1]. Левому потомку каждого узла соответствует подотрезок [a, floor((a+b) / 2)], а правому [ceil((a+b) / 2), b].

Для построения дерева потребуется инициализировать $O(N)$ узлов. Инициализация каждого узла выполняется за константное время.

Для запроса суммы на отрезке или модификации какого-либо элемента исходного массива будем опускаться из корня в соответствующие запрошенным отрезкам узлы. Высота такого дерева есть $O(logN)$, а значит и любой подобный запрос будет выполняться не дольше чем $O(logN)$.

Учитывая всё вышесказанное, окончательная сложность работы алгоритма есть $O(N + M \cdot logN)$.

Код решения (C++)

```
#pragma comment(linker,"/STACK:256000000")
#define _CRT_SECURE_NO_WARNINGS
#define _USE_MATH_DEFINES

#include <iostream>
#include <cmath>
#include <algorithm>

using namespace std;

typedef struct KV {
    KV *left;
    KV *right;
    int l;
    int r;
    int value;
} KV;

KV *build(KV *ar, int l, int r)
{
    if (l == r)
    {
        KV *kv = new KV(ar[l]);
        kv->left = NULL;
        kv->right = NULL;

        return kv;
    }

    int m = (l + r) / 2;

    KV *left = build(ar, l, m);
    KV *right = build(ar, m + 1, r);

    KV *kv = new KV;
    kv->l = l;
    kv->r = r;
    kv->value = left->value + right->value;
    kv->left = left;
    kv->right = right;

    return kv;
}

void set(KV *el, int idx, int val)
{
    int l = el->l;
```

```
    int r = el->r;
    int m = (l + r) / 2;

    if (l == r)
    {
        el->value = val;

        return;
    }

    if (idx <= m)
        set(el->left, idx, val);
    else
        set(el->right, idx, val);

    el->value = el->left->value + el->right->value;
}

int sum(KV *el, int l, int r)
{
    int m = (el->l + el->r) / 2;

    if (l == el->l && r == el->r)
        return el->value;

    if (r <= m)
        return sum(el->left, l, r);
    else if (l > m)
        return sum(el->right, l, r);
    else
        return sum(el->left, l, m) + sum(el->right, m + 1, r);
}

int main()
{
    int n, m;
    KV ar[100000];

    char str[2];
    int a, b;

    scanf("%d%d\n", &n, &m);

    for (int i = 0; i < n; i++)
    {
        ar[i].l = i;
        ar[i].r = i;
        scanf("%d\n", &ar[i].value);
    }

    KV *root = build(ar, 0, n - 1);

    for (int i = 0; i < m; i++)
    {
        scanf("%s %d %d\n", str, &a, &b);
        if (str[0] == 'S')
```

```
            printf("%d\n", sum(root, a, b));
        else
            set(root, a, b);
    }


    return 0;
}
```