Программирование, заключительный этап, 11 класс

Задача 1 (8 баллов)

Условие

Для заданного в 10-й с/с дробного числа требуется определить разницу длин целой и дробной части этого числа в 5-й с/с при переводе с точностью до 8 разрядов.
Входные данные: положительное число в 10-й с/с, целая и дробная часть которого не превышают 8 разрядов каждая. Целая и дробная части разделяются точкой. Дробная часть может отсутствовать.
Выходные данные: неотрицательное целое число - разница между количеством разрядов целой и дробной части в пятеричной системе счисления.

Пример

Входные данные	Выходные данные	Примечание
8.4	1	$8.4_{10}=13.2_5$, количество разрядов целой части на 1 больше, чем дробной
1.5	7	$1.5_{10}=1.(2)_5=1.22222222_5$, в 5-й с/с дробь не имеет конечного представления, поэтому ограничиваем длину 8 дробными разрядами

Проверочные тесты

Входные данные	Ожидаемый результат
8.4	1
1.5	7
2.2	0
222222.1	0
1	1
10000000.2	10
99999999.9	4

Пример решения

```
def perev(x):
      x1 = int(x)
      x2 = float('0.' + str(float(x)).split('.')[-1])
      s = ''
      while x1 != 0:
      s = str(x1 % 5) + s
      x1 //= 5
      sd = ''
      i = 0
      while x2 != 0 and i != 8:
      x2 = x2 * 5
      sd += str(int(x2))
      x2 = float('0.' + str(float(x2)).split('.')[-1])
      i += 1
      return s + '.' + sd


ans = perev(float(input())).split('.')
print(abs(len(ans[0]) - len(ans[-1])))
```

Задача 2 (12 баллов)

Условие

Требуется разделить заданный текст на английском языке на 3 группы слов по их первой букве так, чтобы количество слов в группах отличалось на минимальную величину. Группы могут формироваться только на основе алфавитного порядка латинских букв и задаются в виде диапазонов, например a-i, j-o, p-z.

Входные данные: в первой строке записано число N, не превышающее 10^6 - количество строк в тексте. Далее записано N строк текста, в каждой из которых не более 1000 символов. Текст состоит из латинских букв разного регистра, пробелов и знаков препинания ".", ",", "!", "?", ":". После каждого знака препинания обязательно ставится пробел либо строка заканчивается. Переносы слов с одной строки на другую отсутствуют.

Выходные данные: одно число - величина, на которую отличается количество слов в группе, где их больше всего, от количества слов в группе, где их меньше всего.

Пример

Входные данные	Выходные данные	Примечание
3 Above all zoo the blue bird soars. Hippopotamus hopes high honey hotel?	2	Слова можно разбить на 3 группы: 1-я - 4 слова на буквы a и b; 2-я - 5 слов на букву h; 3-я - 3 слова на буквы s-z.
2 A a a a, o. O o o! A?	5	Слова можно разбить на 2 группы: 1-я - все слова на букву a; 2-я - все слова на букву o; 3-я - пустая.

Проверочные тесты

Входные данные	Ожидаемый результат
3 Above all zoo the blue bird soars. Hippopotamus hopes high honey hotel?	2
2 A a a a, o. O o o! A?	5
1 a b c	0
2 x, y z	0
2 a a, a A? A! A.	6

6 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	2
4 Aa Ab, ba - ab aa a b b b: aaa.	6
4 Aa Ab, ba - ab aa a b b b: zzz.	4

Пример решения

```
#include <bits/stdc++.h>

using namespace std;

int main() {
	ios_base::sync_with_stdio(0);
	cin.tie(0);
	cout.tie(0);
	const int SIZ = 27;
	int a[SIZ];
	memset(a, 0, sizeof(a));
	int n;
	cin >> n;
	string s;
	while (cin >> s) {
	if ('a' <= s[0] && s[0] <= 'z') {
		++a[s[0] - 'a' + 1];
	}
	else if ('A' <= s[0] && s[0] <= 'Z') {
		++a[s[0] - 'A' + 1];
	}
	}
	for (int i = 1; i < SIZ; ++i) {
	a[i] += a[i - 1];
	}
	int mn = a[SIZ - 1];
	for (int i = 0; i < SIZ; ++i) {
	for (int j = i; j < SIZ; ++j) {
		mn = min(mn, max(max(a[i], a[j] - a[i]), a[SIZ - 1] - a[j]) -
min(min(a[i], a[j] - a[i]), a[SIZ - 1] - a[j]));
	}
```

```
        }
        cout << mn;
        return 0;
}
```

Задача 3 (16 баллов)

Условие

Формальной верификацией (проверкой) называется математическое доказательство соответствия или несоответствие предмета верификации его формальному описанию.

Известно, что целочисленная переменная обычно занимает ячейку памяти фиксированного размера, а значит, имеет заранее предопределённый, установленный её типом данных, диапазон допустимых значений, выход за пределы которого приведёт к *переполнению*.

В рамках данной задачи требуется определить, какие значения могут принимать переменные некоторой программы, чтобы в процессе её работы не произошло ни одного переполнения.

Входные данные: в первой строке натуральное число N, не превышающее 10, - количество переменных. Далее N строк, в которых через пробел записано имя переменной в виде одной заглавной латинской буквы, и два целых числа (по модулю не превышают 10^6) - минимальное и максимальное значения, которые определяются её типом данных. Затем в следующей строке записано натуральное число M, не превышающее 100 - количество операций над переменными. Далее в M строках записаны выражения вида A = B + K, где A и B - имена переменных, а K - число, не превышающее по модулю 10^6. Допустимы две операции: сложение и вычитание.

Выходные данные: вывести N строк, где для каждой переменной через пробел указать её имя и диапазоны значений, которые могут быть ей присвоены перед первой операцией присваивания, чтобы гарантированно не произошло ни одного переполнения. Переменные вывести в соответствии с алфавитным порядком их имён.

Пример

Входные данные	Выходные данные
3 A -1000 1000 B 0 1000 C 1 2000 2 B = A + 100 C = B - 200	A 101 900 B 0 1000 C 1 2000
2 A 0 255 B 0 255 5 B = A + 100 B = B - 200 A = B - 1 B = A - 1 A = B - 1	A 103 155 B 0 255

Проверочные тесты

Входные данные	Ожидаемый результат
3 A -1000 1000 B 0 1000 C 1 2000 2 B = A + 100 C = B - 200	A 101 900 B 0 1000 C 1 2000

2 A 0 255 B 0 255 5 B = A + 100 B = B - 200 A = B - 1 B = A - 1 A = B - 1	A 103 155 B 0 255
1 A 0 10 3 A = A + 1 A = A + 1 A = A + 1	A 0 7
1 A 0 10 2 A = A + 1 A = A - 1	A 0 9
3 A 0 10 B 0 10 C 0 10 5 B = A + 1 C = B + 1 A = C + 1 B = A + 1 C = B + 1	A 0 5 B 0 10 C 0 10
3 A 0 100 B 0 200 C 0 300 5 B = A + 1 B = C + 10 A = C + 100 C = B + 20 C = A + 200	A 0 100 B 0 200 C 0 0

3 C 0 300 B 100 300 A 200 300 5 B = A - 1 B = C - 10 A = C - 100 C = B - 20 C = A - 200	A 200 300 B 100 300 C 300 300
8 A 0 1 B 1 2 C 2 3 D 3 4 E 4 5 F 5 6 G 6 7 H 7 8 7 B = A + 1 C = B + 1 D = C + 1 E = D + 1 F = E + 1 G = F + 1 H = G + 2	A 0 0 B 1 2 C 2 3 D 3 4 E 4 5 F 5 6 G 6 7 H 7 8

Пример решения

```
#include <bits/stdc++.h>

using namespace std;

using ll = long long;

int main() {
      ios_base::sync_with_stdio(false);
      cin.tie(nullptr);
      cout.tie(nullptr);
      ll n;
      cin >> n;
      map<char, pair<ll, ll>> mp;
      map<char, pair<char, ll>> queries;
      map<char, pair<ll, ll>> ans;
      for (ll i = 0 ; i < n ; i++) {
      pair<ll, ll> pr;
      char a;
      cin >> a;
      cin >> pr.first >> pr.second;
      mp[a] = pr;
```

```
	ans[a] = pr;
	queries[a] = {a, 0};
	}
	ll m;
	cin >> m;
	for (ll i = 0 ; i < m ; i++) {
	char fst, scd;
	char c;
	ll num;
	cin >> fst;
	cin >> c;
	cin >> scd;
	cin >> c;
	cin >> num;
	if (c == '+') {
		queries[fst] = {queries[scd].first, queries[scd].second + num};
	} else {
		queries[fst] = {queries[scd].first, queries[scd].second - num};
	}
	ans[queries[fst].first].first = max(ans[queries[fst].first].first,
-queries[fst].second + mp[fst].first);
	ans[queries[fst].first].second = min(ans[queries[fst].first].second,
mp[fst].second - queries[fst].second);
	}
	for (const auto &el : ans) {
	cout << el.first << ' ' << el.second.first << ' ' << el.second.second
<< '\n';
	}
	return 0;
}
```

Задача 4 (20 баллов)

Условие

Карта Карно - графический способ представления логической функции, составляемый для формирования минимизированной функции в аналитическом виде.
Для логической функции от четырёх переменных f(a, b, c, d) карта составляется следующим образом:

ab cd	00	01	11	10
00	f(0,0,0,0)	f(0,0,0,1)	f(0,0,1,1)	f(0,0,1,0)
01	f(0,1,0,0)	f(0,1,0,1)	f(0,1,1,1)	f(0,1,1,0)
11	f(1,1,0,0)	f(1,1,0,1)	f(1,1,1,1)	f(1,1,1,0)
10	f(1,0,0,0)	f(1,0,0,1)	f(1,0,1,1)	f(1,0,1,0)

После составления карты в ней выделяют "склейки" - прямоугольные области, удовлетворяющие двум условиям:

- все значения истинны;
- размер области равен 2^n, где n - любое натуральное число.

При этом считают, что первый и последний столбец, а также первая и последняя строки расположены "рядом", то есть в них также можно формировать склейки.
Цель формирования склеек - выделить как можно меньшее их число, для этого склейки должны иметь наибольший размер и могут накладываться друг на друга.

Для логической функции от четырёх переменных требуется составить карту Карно, в которой указать разными цифрами формируемые склейки: самые большие для этой функции (по 8 элементов) - цифрой 4, следующие по размеру (по 4 элемента) - цифрой 3, и т.д.
Входные данные: 16 строк, составляющие полную таблицу истинности функции. В каждой строке через пробел записаны значения переменных a, b, c, d и значение функции f в виде нулей и единиц (0 - значение ложно, 1 - значение истинно).
Выходные данные: матрица из 4 строк по 4 цифры, записанных через пробел и соответствующих искомым значениям. Цифры могут принимать значения от 0 до 4.

Пример

Входные данные	Выходные данные	Примечание
0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 1 0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 1 0 0 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 0 1 1 0 1 0 1 1 1 0 0 1 1 1 1 0	0 3 3 0 3 3 3 3 0 0 0 0 0 3 3 2	Для заданной функции выделяются склейки во 2-й строке и в квадрате в первой и последней строках по 4 элемента (обозначены цифрой 3), а также склейка из двух элементов в конце 4-й строки. Поскольку она накладывается на предыдущую склейку, то только второй элемент в ней обозначен цифрой 2.

0 0 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 1 0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1	3 3 0 0 4 4 4 4 4 4 4 4 0 0 0 0	Для данной функции выделяется склейка во 2-й и 3-й строках, она состоит из 8 элементов и обозначается цифрой 4. Также выделяется квадрат из 4-х элементов (первые 2 в 1-й и 2-й строках), он накладывается на большую склейку, поэтому только его половина отмечена цифрой 3.

Проверочные тесты

Входные данные	Ожидаемый результат
0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 1 0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 1 0 0 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 0 1 1 0 1 0 1 1 1 0 0 1 1 1 1 0	0 3 3 0 3 3 3 3 0 0 0 0 0 3 3 2
0 0 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 1 0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1	3 3 0 0 4 4 4 4 4 4 4 4 0 0 0 0

0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 1 0 0 1 1 1 1 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 1 1 0 0 1 0 1 0 0 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 1 0 0 1 1 1 1 0	3 3 3 3 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 0 0 0 1 1 0 0 1 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 1 1 1 0 1 0 1 1 1 0 0 1 1 1 1 0	3 0 0 0 3 0 0 0 3 0 0 0 3 0 0 0

0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 1 0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1	0 0 0 0 4 4 4 4 4 4 4 4 0 0 0 0
0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 0 0 0 1 1 0 0 1 1 1 0 1 0 0 1 0 1 1 0 1 1 0 0 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1	2 0 2 2 0 0 0 0 3 3 3 3 3 3 0 0
0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1	2 0 3 3 0 0 0 0 3 3 3 3 0 0 3 3

0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 1 0 1 1 1 0 1 0 0 0 1 1 0 0 1 0 1 0 1 0 1 1 0 1 1 0 1 1 0 0 1 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1	4 0 0 4 4 0 0 4 4 0 2 4 4 0 0 4
0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 0 0 1 0 1 0 1 0 1 1 0 1 1 0 1 1 0 0 0 1 1 0 1 1 1 1 1 0 0 1 1 1 1 1	3 0 0 3 0 3 3 0 0 3 3 0 3 0 0 3

Пример решения

```
t = [[0 for i in range(4)] for j in range(4)]
for i in range(16):
      a, b, c, d, f = map(int, input().split())
      t[a*2+(1-a*2)*b+a][c*2+(1-c*2)*d+c] = f

mp = [[0 for i in range(4)] for j in range(4)]
for i in range(4):
      for j in range(4):
      if t[i][j] == 1:
            mp[i][j] = max(1, mp[i][j])
      if t[i][j] == 1 and t[i][j-1] == 1:
            mp[i][j] = max(2, mp[i][j])
            mp[i][j-1] = max(2, mp[i][j-1])
      if t[i][j] == 1 and t[i-1][j] == 1:
            mp[i][j] = max(2, mp[i][j])
            mp[i-1][j] = max(2, mp[i-1][j])
      if all(t[i][j-w] for w in range(4)):
            for w in range(4):
                  mp[i][j-w] = max(3, mp[i][j-w])
      if all(t[i-w][j] for w in range(4)):
```

```
            for w in range(4):
                  mp[i-w][j] = max(3, mp[i-w][j])
      if t[i][j] and t[i-1][j-1] and t[i][j-1] and t[i-1][j]:
            mp[i][j] = max(3, mp[i][j])
            mp[i][j-1] = max(3, mp[i][j-1])
            mp[i-1][j] = max(3, mp[i-1][j])
            mp[i-1][j-1] = max(3, mp[i-1][j-1])
      if all([t[i][j-w] for w in range(4)]+[t[i-1][j-w] for w in range(4)]):
            for w in range(2):
                  for ww in range(4):
                        mp[i-w][j-ww] = 4
      if all([t[i-w][j] for w in range(4)]+[t[i-w][j-1] for w in range(4)]):
            for ww in range(2):
                  for w in range(4):
                        mp[i-w][j-ww] = 4
for i in mp:
      print(*i)
```

Задача 5 (20 баллов)

Условие

Для передачи пакетов в глобальной компьютерной сети используются специальные устройства - маршрутизаторы. Они подключены к нескольким линиям связи и на основе содержащихся в их памяти таблиц маршрутизации определяют, по какому из направлений передавать приходящие пакеты.
IPv4 - 4-я и первая, получившая широкое распространение, версия протокола IP (Internet Protocol). Адреса в этой версии представляют собой 4 октета - блока по 8 бит. В человекочитаемом представлении адреса принято записывать в форме 4-х десятичных чисел от 0 до 255, разделённых точками.
Поскольку маршрутизатор имеет несколько интерфейсов (физических разъёмов, куда подключены кабели связи), в таблице маршрутизации задано соответствие каждого интерфейса той или иной сети. Сеть определяется парой из IP-адреса и битовой маски (в которой N старших бит заполнены единицами, а 32-N младших заполнены нулями). Для проверки того, можно ли маршрутизировать пакет в какую-либо сеть, битовая маска применяется к адресу сети и адресу узла назначения из пакета с помощью побитового И. Если получившийся результат совпадает, то пакет можно отправить на соответствующий интерфейс.

Протокол управления передачей TCP (transmission control protocol) - протокол более высокого уровня по сравнению с IP, предназначенный для доставки данных в неизменном виде. Для решения этой задачи в нём применяется ряд принципов, один из которых - подтверждение получения данных. Для подтверждения от получателя к отправителю направляется специальный пакет с признаком ACK, в котором указан диапазон полученных байтов.
При этом для исключения перегрузок сети при передаче больших объёмов данных применяются "окна" передачи. Окном называется максимальный объём данных, который может быть отправлен от отправителя получателю без подтверждения получения. По мере подтверждения получения данных окно смещается вправо и отправляются новые данные.

Входные данные: в первой строке записано натуральное число N, не превышающее 100 - количество маршрутизаторов. Далее записано N таблиц маршрутизации в следующем формате: в первой строке число N1 - количество маршрутов в таблице, далее N1 строк, где через пробел указаны адрес сети, маска (целое число от 1 до 32), номер маршрутизатора, куда будет отправлен пакет (номера считаются с 1 по порядку описания), и время передачи до следующего маршрутизатора в миллисекундах.
После описания таблиц маршрутизации в следующей строке записаны через пробел адрес узла-отправителя, номер маршрутизатора, к которому он подключён, адрес узла-получателя, номер маршрутизатора, к которому подключён получатель, и объём данных (в байтах), который требуется передать.
Выходные данные: минимальное время передачи в миллисекундах.
Примечание: считать, что между получением каждого пакета и отправкой подтверждения с конечного устройства всегда проходит 1 мс.

В рамках данной задачи считать, что потери отсутствуют и все отправленные пакеты достигают получателя, а также что размер окна фиксированный и составляет 10240 байт. Также следует пренебречь временем передачи данных между конечными узлами и ближайшими маршрутизаторами.

Пример

Входные данные	Выходные данные
5 3 10.0.0.0 8 2 10 20.0.0.0 8 3 10 10.0.0.0 8 5 10 1 10.0.0.0 8 4 10 1 30.0.0.0 8 4 10 1 40.0.0.0 8 1 10 1 10.0.0.0 8 4 20 40.1.1.1 1 10.1.1.1 4 102400	310

Примечание: 102400 = 10240*10, значит, для передачи данных потребуется 10 итераций (сдвигов окна передачи). Каждые 10 Кбайт будут проходить следующую цепочку: отправка с первого маршрутизатора на второй - 10мс, затем со 2-го на 4-й ещё 10 мс, затем 1 мс задержки на отправку подтверждения и сама передача подтверждения напрямую с 4-го маршрутизатора на 1-й - ещё 10 мс. Таким образом, на передачу каждых 10 Кбайт будет уходить 31 мс.

Проверочные тесты

Входные данные	Ожидаемый результат

5 3 10.0.0.0 8 2 10 20.0.0.0 8 3 10 10.0.0.0 8 5 10 1 10.0.0.0 8 4 10 1 30.0.0.0 8 4 10 1 40.0.0.0 8 1 10 1 10.0.0.0 8 4 20 40.1.1.1 1 10.1.1.1 4 102400	310
2 1 200.0.0.0 16 2 10 1 100.0.0.0 16 1 10 100.0.0.1 1 200.0.0.1 2 1	21
2 1 200.0.0.0 16 2 100 1 100.0.0.0 16 1 200 100.0.0.1 1 200.0.0.1 2 10240	301
4 2 20.0.0.0 16 2 10 30.0.0.0 16 2 10 1 30.0.0.0 8 3 20 2 40.0.0.0 24 4 30 10.0.0.0 24 4 30 1 10.0.0.0 16 1 40 10.0.0.1 1 30.0.0.1 3 9999	101
4 2 20.0.0.0 16 2 10 30.0.0.0 16 2 10 2 30.0.0.0 8 3 20 10.0.0.0 16 1 10 2 40.0.0.0 24 4 30 10.0.0.0 24 4 30 1 10.0.0.0 16 2 40 10.0.0.1 1 30.0.0.1 3 9999	111

4 3 20.0.0.0 16 2 10 30.0.0.0 16 2 10 30.0.0.0 24 3 15 2 30.0.0.0 8 3 20 10.0.0.0 16 1 10 3 40.0.0.0 24 4 30 10.0.0.0 24 4 30 10.0.0.0 8 1 15 1 10.0.0.0 16 2 40 10.0.0.1 1 30.0.0.1 3 9999	31
6 2 10.2.0.0 16 2 11 10.6.0.0 16 6 11 3 10.0.0.0 8 3 12 10.4.0.0 16 4 12 10.5.0.0 16 4 12 1 10.0.0.0 8 4 13 1 10.5.0.0 16 5 14 2 10.6.0.0 16 6 15 10.2.0.0 16 6 15 2 10.2.0.0 16 2 16 10.4.0.0 16 4 16 10.2.0.1 2 10.5.0.1 5 1000	58
4 2 192.168.0.0 16 2 20 192.168.128.0 18 3 10 1 192.168.128.0 17 4 10 1 192.168.128.0 24 4 10 1 192.168.0.0 16 1 30 192.168.1.1 1 192.168.193.5 4 1024	61

4 2 20.0.0.0 16 2 10 30.0.0.0 16 2 10 1 30.0.0.0 8 3 20 2 40.0.0.0 24 4 30 10.0.0.0 24 4 30 1 10.0.0.0 16 1 40 10.0.0.1 1 30.0.0.1 3 102400	1010
4 2 20.0.0.0 16 2 10 30.0.0.0 16 2 10 1 30.0.0.0 8 3 20 2 40.0.0.0 24 4 30 10.0.0.0 24 4 30 1 10.0.0.0 16 1 40 10.0.0.1 1 30.0.0.1 3 92161	1010

Пример решения

```
def to8(stroka: str):
      while len(stroka) < 8:
      stroka = '0' + stroka
      return stroka


class Route:
      def __init__(self, net: str, mask: str, next: str, t: str):
      net1 = []
      for i in net.split('.'):
            net1.extend(list(to8(bin(int(i))[2:])))

      self.t = int(t)
      self.net = int(''.join(net1))
      self.next = int(next) - 1
      self.mask = ([1] * (int(mask)) + [0] * (32 - (int(mask))))

      def check(self, x: str):
      net = []
      for i in x.split('.'):
            net.extend(list(to8(bin(int(i))[2:])))
      for i in range(32):
            net[i] = str(int(net[i]) * self.mask[i])
      return int(''.join(net)) == self.net


def obxod(start: int, stroka: str, net: str, lis: list, t: int, to_id):
```

```
    ans = None
    if start == to_id:
    return t
    stroka = stroka + str(start)
    for route in lis[start]:
    if str(route.next) not in stroka and route.check(net):
        x = obxod(route.next, stroka, net, lis, t + route.t, to_id)
        if not (x is None) and (ans is None or ans > x):
            ans = x

    return ans


lis = []
n = int(input())
for i in range(n):
    lis.append([])
    for j in range(int(input())):
    net, mask, next, t, = input().split()
    lis[-1].append(Route(net, mask, next, t))

from_net, from_id, to_net, to_id, v = input().split()
from_id, to_id = int(from_id) - 1, int(to_id) - 1
v = int(v)
ans = obxod(from_id, '', to_net, lis, 0, to_id) + obxod(to_id, '',
from_net, lis, 0, from_id)
ans += 1
if v % 10240:
    ans *= (v // 10240 + 1)
else:
    ans *= v // 10240
print(ans)
```

Задача 6 (24 баллов)

Условие

В реальных компьютерных сетях также используется понятие максимальной единицы передачи (MTU, maximum transmission unit) - размер блока данных, который может быть передан на отрезке сети между двумя устройствами. Этот размер зависит от ограничений используемой технологии передачи данных на нижнем уровне.

В данной задаче требуется составить оптимальный план передачи пакетов с учётом MTU между разными маршрутизаторами и вычислить минимальное время передачи на его основе.

Нужно иметь в виду, что окно передачи работает по "скользящему" принципу, то есть после получения подтверждения доставки первых байтов окна оно сдвигается вправо на размер доставленной информации.

Входные данные: в первой строке записано натуральное число N, не превышающее 100 - количество маршрутизаторов. Далее записано N таблиц маршрутизации в следующем формате: в первой строке число N1 - количество маршрутов в таблице, далее N1 строк, где через пробел указаны адрес сети, маска (целое число от 1 до 32), номер маршрутизатора, куда будет отправлен пакет (номера считаются с 1 по порядку описания), время передачи до следующего маршрутизатора в миллисекундах и MTU в байтах.

После описания таблиц маршрутизации в следующей строке записаны через пробел адрес узла-отправителя, номер маршрутизатора, к которому он подключён, адрес узла-получателя, номер маршрутизатора, к которому подключён получатель, и объём данных (в байтах), который требуется передать.

Выходные данные: минимальное время передачи в миллисекундах.

Примечание: считать, что между получением каждого пакета и отправкой подтверждения с конечного устройства всегда проходит 1 мс.

Примечание 2: считать, если объём данных, который требуется отправить с одного маршрутизатора на другой, превышает MTU, то данные делятся на пакеты, равные максимальной единице передачи или меньше, и отправляются с задержкой в 1 мс.

В рамках данной задачи считать, что потери отсутствуют и все отправленные пакеты достигают получателя, а также что размер окна фиксированный и составляет 10240 байт. Также следует пренебречь временем передачи данных между конечными узлами и ближайшими маршрутизаторами и размером пакета ACK (считать, что он меньше любого MTU).

Пример

Входные данные	Выходные данные
5 3 10.0.0.0 8 2 10 100 20.0.0.0 8 3 10 100 10.0.0.0 8 5 10 10240 1 10.0.0.0 8 4 10 100 1 30.0.0.0 8 4 10 100 1 40.0.0.0 8 1 10 100 1 10.0.0.0 8 4 20 10240 40.1.1.1 1 10.1.1.1 4 102400	410

Примечание: если отправлять пакеты с 1-го маршрутизатора на 5-й, то время передачи каждых 10240 байт с учётом подтверждения займёт 41 мс. Если же отправлять с 1-го через 2-й, то 10240 байт окна передачи будут делиться на 103 блока и суммарное время доставки для первого окна увеличится до 133 мс, а для всего объёма данных - более 1000 мс.

Проверочные тесты

Входные данные	Ожидаемый результат
5 3 10.0.0.0 8 2 10 100 20.0.0.0 8 3 10 100 10.0.0.0 8 5 10 10240 1 10.0.0.0 8 4 10 100 1 30.0.0.0 8 4 10 100 1 40.0.0.0 8 1 10 100 1 10.0.0.0 8 4 20 10240 40.1.1.1 1 10.1.1.1 4 102400	410
2 1 200.0.0.0 16 2 10 999999 1 100.0.0.0 16 1 10 999999 100.0.0.1 1 200.0.0.1 2 1	21
2 1 200.0.0.0 16 2 100 5000 1 100.0.0.0 16 1 200 5000 100.0.0.1 1 200.0.0.1 2 10240	303
4 2 20.0.0.0 16 2 10 9999 30.0.0.0 16 2 10 9999 1 30.0.0.0 8 3 20 9999 2 40.0.0.0 24 4 30 9999 10.0.0.0 24 4 30 9999 1 10.0.0.0 16 1 40 9999 10.0.0.1 1 30.0.0.1 3 9999	101

4 2 20.0.0.0 16 2 10 9000 30.0.0.0 16 2 10 9000 2 30.0.0.0 8 3 20 9000 10.0.0.0 16 1 10 9000 2 40.0.0.0 24 4 30 9000 10.0.0.0 24 4 30 9000 1 10.0.0.0 16 2 40 9000 10.0.0.1 1 30.0.0.1 3 9999	112
4 3 20.0.0.0 16 2 10 999999 30.0.0.0 16 2 10 999999 30.0.0.0 24 3 15 999999 2 30.0.0.0 8 3 20 999999 10.0.0.0 16 1 10 999999 3 40.0.0.0 24 4 30 999999 10.0.0.0 24 4 30 999999 10.0.0.0 8 1 15 999999 1 10.0.0.0 16 2 40 999999 10.0.0.1 1 30.0.0.1 3 9999	31
6 2 10.2.0.0 16 2 11 100 10.6.0.0 16 6 11 100 3 10.0.0.0 8 3 12 100 10.4.0.0 16 4 12 100 10.5.0.0 16 4 12 100 1 10.0.0.0 8 4 13 100 1 10.5.0.0 16 5 14 100 2 10.6.0.0 16 6 15 100 10.2.0.0 16 6 15 100 2 10.2.0.0 16 2 16 100 10.4.0.0 16 4 16 100 10.2.0.1 2 10.5.0.1 5 1000	67

4 2 192.168.0.0 16 2 20 2000 192.168.128.0 18 3 10 3000 1 192.168.128.0 17 4 10 4000 1 192.168.128.0 24 4 10 5000 1 192.168.0.0 16 1 30 6000 192.168.1.1 1 192.168.193.5 4 1024	61
4 2 20.0.0.0 16 2 10 10240 30.0.0.0 16 2 10 10240 1 30.0.0.0 8 3 20 10240 2 40.0.0.0 24 4 30 10240 10.0.0.0 24 4 30 10240 1 10.0.0.0 16 1 40 10240 10.0.0.1 1 30.0.0.1 3 102400	1010
4 2 20.0.0.0 16 2 10 10240 30.0.0.0 16 2 10 10240 1 30.0.0.0 8 3 20 10240 2 40.0.0.0 24 4 30 10240 10.0.0.0 24 4 30 10240 1 10.0.0.0 16 1 40 10240 10.0.0.1 1 30.0.0.1 3 92161	1010
4 2 2.0.0.0 16 2 10 100 4.0.0.0 16 2 10 100 3 1.0.0.0 16 2 10 100 3.0.0.0 16 2 10 100 4.0.0.0 16 2 10 100 2 1.0.0.0 16 2 10 100 4.0.0.0 16 2 10 50 2 1.0.0.0 16 3 10 100	70

3.0.0.0 16 3 10 100 1.0.0.1 1 4.0.0.1 4 999	
4 2 2.0.0.0 16 2 10 1024 4.0.0.0 16 2 10 1024 3 1.0.0.0 16 2 10 1024 3.0.0.0 16 2 10 1024 4.0.0.0 16 2 10 1024 2 1.0.0.0 16 2 10 1024 4.0.0.0 16 2 10 1024 2 1.0.0.0 16 3 10 1024 3.0.0.0 16 3 10 1024 1.0.0.1 1 4.0.0.1 4 102400	619

Пример решения

```
def to8(stroka: str):
      while len(stroka) < 8:
      stroka = '0' + stroka
      return stroka


class Route:
      def __init__(self, net: str, mask: str, next: str, t: str, mtu):
      net1 = []
      for i in net.split('.'):
            net1.extend(list(to8(bin(int(i))[2:])))

      self.t = int(t)
      self.net = int(''.join(net1))
      self.next = int(next) - 1
      self.mask = ([1] * (int(mask)) + [0] * (32 - (int(mask))))
      self.mtu = int(mtu)

      def check(self, x: str):
      net = []
      for i in x.split('.'):
            net.extend(list(to8(bin(int(i))[2:])))
      for i in range(32):
            net[i] = str(int(net[i]) * self.mask[i])
      return int(''.join(net)) == self.net


lis = []
n = int(input())
for i in range(n):
      lis.append([])
```

```
      for j in range(int(input())):
      net, mask, next, t, mtu = input().split()
      lis[-1].append(Route(net, mask, next, t, mtu))

from_net, from_id, to_net, to_id, v = input().split()
from_id, to_id = int(from_id) - 1, int(to_id) - 1
v = int(v)

k = 10240


def obxod(start, to_net, to_id, v, stroka='', t=0, vi=k, ti=0):
      ans = None
      if start == to_id:
      return t + ti
      stroka = stroka + str(start)
      for route in lis[start]:
      if str(route.next) in stroka:
            continue
      if not route.check(to_net):
            continue
      a = v // k
      a += 1 if v % k else 0
      ti1 = ti + vi // route.mtu
      ti1 += 1 if vi % route.mtu else 0
      ti1 -= 1
      x = obxod(route.next, to_net, to_id, v, stroka, t + route.t * a,
min(vi, route.mtu), ti1)
      if not (x is None) and (ans is None or ans > x):
            ans = x
      return ans


a = v // k
a += 1 if v % k else 0

print(obxod(from_id, to_net, to_id, v, ) + obxod(to_id, from_net, from_id,
v, vi=1) + a)
```

КРИТЕРИИ ОЦЕНИВАНИЯ ОЛИМПИАДНЫХ ЗАДАНИЙ ЗАКЛЮЧИТЕЛЬНОГО ЭТАПА ОЛИМПИАДЫ ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

Максимальная сумма баллов за выполненные задания варианта – 100.

8 - 11 классы

Распределение баллов по заданиям:

Номер задачи	1	2	3	4	5	6 Ситуационная задача
Баллы	8	12	16	20	20	24

Балл за каждую задачу проставляется в соответствии с долей выполнения участником задания. Шаг оценивания каждого задания - 1 балл. Доля выполнения вычисляется автоматической системой тестирования на основе комплекта установленных для задания тестов.