Предмет	Класс	Дата	Время начала	Время окончания
информатика	***9-11***		***10-00***	***13-55***

Разбор решений.

Все истории и персонажи где-то есть, любая случайность — настоящее совпадение.

Задача 1 «Несчастливый билет» (10 баллов).

Как и всем гениям, юному программисту Васе Петечкину иногда в голову приходят странные мысли. Вася боится несчастливых билетов и, каждый раз перед поездкой в автобусе, он подсчитывает вероятность их выпадения.

Назовем автобусный билет несчастливым, если сумма цифр его шестизначного номера делится на 13. Помогите Васе определить количество несчастливых билетов в одном рулоне. Рулон задается номером его первого и последнего билета, нумерация билетов идет в порядке возрастания.

Формат входных данных: Вводится два целых числа N, M ($100000 \le N \le 999999$) — номера первого и последнего билета.

Формат выходных данных: Программа должна вывести одно число – количество несчастливых билетов в рулоне.

Пример:

Входные данные	Выходные данные
900001 900026	3

Решение:

Алгоритм решения состоит из следующих этапов:

1. Чтение данных из файла.
2. Для каждого числа из указанного промежутка чисел:
   2.1.Нахождение суммы цифр.
   2.2.Проверка делимости суммы на 13.
   2.3.Если сумма кратна 13, то увеличить счетчик на единицу.
3. Вывод счетчика в файл.

Поскольку в алгоритме повторяются действия нахождения суммы цифр, то данную подзадачу можно вынести в отдельную функцию.

C

```
void main()
{
//открываем файл и считываем данные
FILE *input, *output;
input = freopen("input.txt", "r", stdin);
long int N, M, a;
```

```
scanf("%li %li", &N, &M);
int k = 0;
        for (long int i = N; i <= M; i++)
        {a = i;
        int c, sum = 0;
        while (a != 0)
        {//находим сумму цифр текущего номера
                c = a % 10;
                sum += c;
                a /= 10;
        }
//если сумма кратна 13, то увеличиваем счетчик
        if (sum % 13 == 0)
                k++;
                }
//открываем файл и помещаем результат
        output = freopen("output.txt", "w", stdout);
        printf("%i", k);
//закрываем оба файла
        fclose(input);
        fclose(output);
}
```

Задача 2 «Стихоплет» (10 баллов)

Васе Петечкину к уроку литературы необходимо приготовить примеры стихотворных текстов с разными видами рифм, но Вася больше любит программировать, чем читать стихи и разбираться в их сложении. Поэтому он решил скачать из интернета всевозможные тексты и сделать программу, которая будет сама анализировать их и выбирать нужные.

Вася определил для простоты, что стихотворный текст – это текст, записанный в 4 строки, объединенных рифмой и имеющих одинаковое количество слогов в рифмуемых строках. Количество слогов в строке равно количеству в ней гласных букв (он помнил, что гласные: **a, e, i, o, u, y, ‘e, ‘u, ‘a,** могут быть строчными или прописными). Рифмой называется совпадающая концовка пар строк (две и более буквы). В учебнике он прочитал, что рифмы бывают смежные (первая строка рифмуется со второй, третья – с четвертой, обозначим ее CCDD), перекрестные (первая строка рифмуется с третьей, вторая – с четвертой, обозначим CDCD) и кольцевые (рифмуются первая и четвертая строки, а также вторая и третья, обозначим CDDC). Помогите Пете написать программу, которая по заданной четверке строк определит, является ли строка стихотворной и укажет тип рифмы и количество слогов в рифмуемых строках. Если текст не рифмуется, то выведет

сообщение «NO» и количество слогов в каждой строке.

Формат входных данных: Из файла вводится четыре строки (длина строки не превосходит 256 символов). Пусть строки англоязычные или на транслите без знаков препинания. Используется только знак «’» - апостроф.

Формат выходных данных: Вывести сообщение «NO», если текст не рифмованный, иначе вывести одно из сообщений CCDD, CDCD, CDDC, CCCC (если рифмуются все 4 строки) и, в любом случае, через тире количество слогов в каждой строке.

Примеры.

Входные данные	Выходные данные
‘A pomn’u chudnoe mgnoven’e Peredo mnoj ‘avilas’ ty Kak mimoletnoe viden’e Kak genij chistoj krasoty	CDCD 9-8-9-8
Who has seen the wind Neither I nor you But when the leaves hang trembling The wind is passing through	NO 6-8-9-7

Решение:

Задача имеет две составные части:

а) нахождение рифмы с определением ее типа,

б) нахождение количества слогов в каждой строке.

Однако обе подзадачи легко будут решены после предварительной подготовки строк. Подготовка подразумевает:

1. Замену всех прописных букв строчными.
2. Замену всех гласных цифрами.
3. Удаление пробелов между словами.

Все эти преобразования не влияют на итог, но упрощают основной алгоритм, т.к. будут применяться с 4 строкам стихотворения.

Все три подготовительные процедуры вынесем в отдельные функции.

1. Функция, назовем ее «`replace`», будет в массиве символов заменять все строчные буквы прописными.

```
char replace(int n, char *s)
{
        int i = 0;
        while (i <= n)
        {
```

```
                if (s[i] >= 'A'&&s[i] <= 'Z')
                {
                        s[i] = s[i] - 'A' + 'a';
                }
                i++;
        }
        return (*s);
};
```

2. Так как всего у нас 9 разновидностей гласных, то заменим их на цифры от 1 до 9. Изменим порядок букв так, чтобы гласные с апострофом шли в том же порядке, что и их твердые аналоги: a-0, e-1, u -2, i -3, o -4, y-5, ‘a -6, ‘e -7, ‘u -8. Заметим, что у нас остались только строчные гласные буквы, т.к. мы выполнили преобразование из пункта 1.

Очевидно, что парные гласные находятся друг от друга на расстоянии 5 символов в получившемся упорядоченном ряду. Будем использовать это наблюдение для сокращения перебора букв. Создадим массив символов glas, размерность которого будет равна 7 – 6 различных гласных и апостроф.

Создадим вспомогательную переменную flag и присвоим ей значение 0. Просматривая символы в строке, будем их сравнивать с символами массива glas. Как только в строке встретится апостроф, заменим его на пробел, а переменной flag присвоим 1.

Если же выбранная в строке буква является гласной (присутствует в массива glas), то заменим ее на символ, соответствующий порядковому номеру гласной в этом массиве. Для этого к порядковому номеру гласной в массиве прибавим код «0», если значения переменной flag не менялось, т.е. равно 0. Если же замена производилась, т.е. данной букве предшествовал апостроф, то заменяем гласную на порядковому номеру гласной в массиве плюс 5 и прибавим код «0».

```
char rep_glas(int n, char *s) {
        char glas[7] = {'a', 'e', 'u', 'i', 'o', 'y','\''};
        int i = 0;
        while (i <= n) {
                int flag = 0;
                for (int j = 0; j < 7; j++) {
                        if (s[i] == glas[j]) {
                                if (j != 6) {
                                        if (flag == 0) s[i] = j + '0';
                                        else s[i] = j + 5 + '0';
                                }
                                else {
                                        s[i] = ' ';
                                        flag = 1;
                                }
```

```
                        }
                }
                i++;
        }
        return (*s);
}
```

3. Далее, удалим все пробелы, т.к. их наличие не влияет на определение рифмы и количества слогов в строке.

Функцию, которая будет выполнять эти действия, назовем del. Ее аргументами, как и в предыдущих двух, являются длина строки и символьный массив из букв этой строки. Результатом работы является символьный массив, не содержащий пробелов.

```
char del(int n, char *s) {
        for (int i = 0; i <= n; i++) {
                if (s[i] == ' ') {
                        for (int j = i; j < n; j++)
                                s[j] = s[j + 1];
                        i = i - 1;
                }
        }
        return(*s);
}
```

Функция просматривает элементы массива. Как только встречается пробел производится сдвиг всех элементов массива на одну позицию. После этого шага продолжается просмотр строки с предыдущего символа.

После сложных подготовительных процедур решение задачи становится довольно простым. Чтобы определить наличие рифмы будем сравнивать поочередно концовки (два последних символа) каждой пары строк. Пары составляем по условию задачи. Эта часть программы имеет 4 оператора сравнения и вывод результата в файл:

```
int main(){
char s1[256],s2[256],s3[256],s4[256];
//ввод данных
{…….}
//определение рифмы
        char *result[4];
        int flag = 0;
        if (flag == 0 && s1[strlen(s1) - 2] == s2[strlen(s2) - 2] && s1[strlen(s1) - 1] ==
s2[strlen(s2) - 1] && s3[strlen(s3) - 2] == s4[strlen(s4) - 2] && s3[strlen(s3) - 1] ==
s4[strlen(s4) - 1] && s2[strlen(s2) - 2] == s3[strlen(s3) - 2] && s2[strlen(s2) - 1] ==
s3[strlen(s3) - 1]) {
                result ="CCCC";
                flag = 1;
```

```
        }
        if (flag == 0 && s1[strlen(s1) - 2] == s2[strlen(s2) - 2] && s1[strlen(s1) - 1] ==
s2[strlen(s2) - 1] && s3[strlen(s3) - 2] == s4[strlen(s4) - 2] && s3[strlen(s3) - 1] ==
s4[strlen(s4) - 1]) {
                result = "CCDD";
                flag = 1;
        }
        if (flag == 0 && s1[strlen(s1) - 2] == s3[strlen(s3) - 2] && s1[strlen(s1) - 1] ==
s3[strlen(s3) - 1] && s2[strlen(s2) - 2] == s4[strlen(s4) - 2] && s2[strlen(s2) - 1] ==
s4[strlen(s4) - 1]) {
                result ="CDCD";
                flag = 1;
        }
        if (flag == 0 && s1[strlen(s1) - 2] == s4[strlen(s4) - 2] && s1[strlen(s1) - 1] ==
s4[strlen(s4) - 1] && s2[strlen(s2) - 2] == s3[strlen(s3) - 2] && s2[strlen(s2) - 1] ==
s3[strlen(s3) - 1]) {
                result ="CDDC";
                flag = 1;
        }
        if (flag == 0) {
                        result ="NO";
                }
```

Осталось найти количество слогов в каждой строке. Оно совпадает с количеством гласных букв в ней. Так как мы произвели замену всех гласных на цифры, то и будем подсчитывать количество встреченных в строке цифр. Т.к. данная операция применяется к каждой из 4-х строк в отдельности, то имеет смысл описать целочисленную функцию подсчета слогов.

```
int vowel(int dl, char *s) //счет количества слогов в строке
{
        int count = 0;
        for (int i = 0; i <= dl; i++) {
                if (s[i] >= '0'&&s[i] <= '9') count++;
        }
        return(count);
}
```

Результат выполнения этой функции для каждой строки выводим в файл через знак тире.

Таким образом, остается только проверить совпадение количества слогов в рифмуемых строках.

Предмет	Класс	Дата	Время начала	Время окончания
информатика	***9-11***		***10-00***	***13-55***

Задача 3. «Колонна» (10 баллов)

Васины одноклассники на уроке физкультуры учились выполнять повороты налево и направо, по команде учителя все дети одновременно повернулись кто-то, как и просил учитель налево, но некоторые, самые рассеянные, повернулись в противоположном направлении. Оказавшись лицом друг к другу дети поняли, что совершили ошибку и сразу развернулись кругом. Эта операция продолжалась до тех пор, все дети не оказались стоять в колонне, видя перед собой затылок одноклассника.

Вся эта суета озадачила Васю, и он решил подсчитать количество пар школьников, которые после команды учителя совершали впоследствии развороты кругом.

Формат входных данных: в файле в первой строке одно число N ($2 \le N \le 1000$) – количество учеников в шеренге. Во второй строке – последовательность из N символов «<» или «>». Символ «<» означает, что ученик, стоящий на соответствующем месте повернулся налево, «>» - соответственно направо.

Формат выходных данных: необходимо вывести либо одно число – количество развернувшихся пар, либо сообщение «NO», если процесс бесконечен.

Пример:

Входные данные	Выходные данные
6 >><<><	7
2 <>	0

Решение:

Заметим, что школьники могут после всех перестановок оказаться в позиции, когда одна часть колонны смотрит налево, а другая направо. Так как в этом состоянии никто из школьников не стоит лицом друг к другу, то построение будем считать оконченным.

Основной идеей решения задачи является просмотр и сравнение соседних элементов строки S. В один проход по строке считаем количество соседний вида “><”. Следующим проходом по этой строке меняем эти символы на противоположные.

Для оптимизации алгоритма и уменьшения количества просмотров строки введем дополнительную строковую переменную, назовем ее temp. Перед каждым просмотром строки будем присваивать этой дополнительной переменной строку S. В основном цикле программы будем сравнивать элементы строки temp. Если обнаружим соседей, смотрящих друг на друга, то меняем на противоположные символ значения в основной строке S, находящиеся на тех же местах, что и найденные в temp символы.

Предмет	Класс	Дата	Время начала	Время окончания
информатика	***9-11***		***10-00***	***13-55***

```
#include <stdio.h>
#include <conio.h>
#include <string>
#include <iostream>
#include <fstream>
using namespace std;
int main()
{	FILE *input, *output;
	input = freopen("input.txt", "r", stdin);
	int n;
	scanf_s("%d\n", &n); //считываем количество учащихся
	char s[1000];
	cin.getline(s, n+1); //считываем строку состояния колонны
	int kol = 0;
	int flag = 1;
	string temp;
	while (flag == 1) {
		flag = 0; temp = s;
		//подсчет количества соседей вида ><
		for (int i = 0; i < n; i++) {
			if (temp[i] == '>'&&temp[i+1] == '<') {
			kol++;
			s[i] = '<';
			s[i+1] = '>';
			flag = 1; //flag будет равен 1, если в строке произведена хоть одна замена
			}
		}
	}
	output = freopen("output.txt", "w", stdout);
	printf_s("%d", kol);
	fclose(input);
	fclose(output);
}
```

Предмет	Класс	Дата	Время начала	Время окончания
информатика	*9-11*		*10-00*	*13-55*

Задача 4. «Лестница» (10 баллов)

Вася Петечкин подошел к платной лестнице. Чтобы наступить на любую ступеньку, нужно заплатить указанную на ней сумму. «Чего только не придумают, чтобы денег собрать» - подумал Вася и научился не только перешагивать на следующую ступеньку, но и перепрыгивать через ступеньку. Осталось только написать программу, которая будет подсчитывать, какая наименьшая сумма понадобится Васе, чтобы добраться до верхней ступеньки.

Формат входных данных: В первой строке файла находится одно число N – количество ступенек (0<N<1000). В следующей строке через пробел записаны N чисел – стоимость каждой соответствующей ступеньки.

Формат выходных данных: целое число соответствующее минимальной сумме, которая понадобится Васе.

Примеры.

Входные данные	Выходные данные
5 1 2 3 4 5	6

Решение:

Пусть a[N] массив стоимостей ступеней. Расширим его, добавив нулевую ступень, как основание лестницы и n+1 – тротуар за лестницей.

Очевидно, что сумма, которую Вася отдаст на N-ой ступеньке, есть сумма, которую он отдал до этого плюс стоимость самой ступеньки. «Сумма, которую он отдал до этого» зависит от того, с какой ступеньки Вася шагает на N-ую — с (N-1)-й или с (N-2)-й. Для того, чтобы сумма оказалась минимальной, шагать Вася должен со ступени с наименьшей стоимостью.

```
#include <stdio.h>
#include <conio.h>
#include <locale.h>
#include <malloc.h>
#include <math.h>
void main() {
	FILE *input, *output;
	input = freopen("input.txt", "r", stdin);

	int n;
	scanf_s("%d\n", &n); //количество ступеней

	int *a = (int*)malloc(sizeof(int)*(n + 2));//создаем массив из цен ступеней
```

```
        for (int i = 1; i<=n; i++)
                scanf_s("%d", &a[i]);
        a[0] = 0;
        int *temp = (int*)malloc(sizeof(int)*(n + 1));//создаем вспомогательный массив
        temp[0] = a[0];
        temp[1] = a[1];

        for (int i = 2; i < n+1; i++)
        {
                temp[i] = a[i]+fmin(temp[i - 1], temp[i - 2]);
        }
        output = freopen("output.txt", "w", stdout);
        int k;
        k = fmin(temp[n - 1], temp[n]);
        printf_s("%d\n", k);
        fclose(input);
        fclose(output);
}
```

Предмет	Класс	Дата	Время начала	Время окончания
информатика	*9-11*		*10-00*	*13-55*

Задача 5 «Игра в классики» (10 баллов)

Вася Петечкин с друзьями играют в классики. Но они, конечно же, придумали свои правила. Записали натуральные числа в бесконечную таблицу (см. пример) и прыгают от одного числа к другому. Чтобы выиграть, надо не только перепрыгивать с числа на число, но и всегда знать, какие числа будут в соседних клетках (сверху, справа, слева и снизу, если таковые имеются). Напишите программу, которая определяет «соседей».

Формат входных данных: Одно натуральное число N (где 1<N<1000) – номер классика, в котором находится игрок в данный момент.
Формат выходных данных: Числа, записанные в соседних с данным классиком клетках, в порядке возрастания. Числа должны разделяться пробелом.

					…
				17	…
			10	18	…
		5	11	19	…
	2	6	12	20	…
1	3	7	13	21	…
	4	8	14	22	…
		9	15	23	…
			16	24	…
				25	…
					…

Пример:

Входные данные	Выходные данные
1	3
7	3 6 8 13
10	11 18

Решение:

Ответить на вопрос задачи можно, если для заданного номера классика определим номер столбца, в котором он находится и номер этого классика в столбце.

Ответить на этот вопрос можно двумя методами: простым перебором чисел в цикле от 1 до N или математически (через арифметическую прогрессию).

```
#include <stdio.h>
#include <conio.h>
#include <locale.h>
```

```
int main() {
	FILE *input, *output;
	input = freopen("input.txt", "r", stdin);

	int n = 0;
	int i = 1;//последний элемент столбца
	int k = 1;//количество элементов в столбце
	scanf_s("%d", &n);
	output = freopen("output.txt", "w", stdout);
	if (n == 1) {
		printf("%d", 3);//обработка исключения – один сосед
	}
	else {
		while (i < n) {
			k += 2;
			i += k;
		}
		if ((n == i) || (n == i - k + 1)) {
			if (n == i) {
				printf("%d%c%d", n - 1, ' ', n + k + 1);//два соседа
			}
			else {
				printf("%d%c%d", n + 1, ' ', n + k + 1);// два соседа
			}
		}
		else {
			printf("%d%c%d%c%d%c%d", n - k + 1, ' ', n - 1, ' ', n + 1, ' ', n + k + 1);
		}
	}

	fclose(input);
	fclose(output);

}
```