Задания муниципального этапа
всероссийской олимпиады школьников по информатике
2015-2016 учебный год

9-11 классы

Время выполнения – 5 часов
Максимальное количество баллов – 400
Максимальное количество баллов за решение одной задачи – 100

Особенности проверки

На проверку принимаются тексты программ! Проверяющий компилирует текст программы и запускает исполняемый файл для каждого теста. Результат работы программы сравнивается с эталонным. Ответ должен полностью совпадать! В том числе по формату и порядку вывода чисел.

Для всех задач имеются папки с файлами. Там содержится
а) несколько тестовых примеров (как в формулировке задачи)
б) по 10 тестов (входные данные и эталонный ответ)
в) решения жюри.

На проверку принимаются только те решения, которые выдают правильные ответы для тестов, приведенных в примерах. **Баллы за прохождение таких тестов не начисляются!**

За прохождение каждого теста из основной группы начисляется 10 баллов.

При проверке задач необходимо соблюдать следующие требования:

а) хотя бы визуально контролировать время выполнения, оно не должно превышать 1 секунду, а программа не должна зависать;

б) проверять текст программы визуально, при этом программы-пустышки, выводящие одинаковый результат независимо от входных данных, выдающие конкретное число на конкретные входные данные (то есть не содержащие как такового алгоритма), оцениваются в 0 баллов!

Просим также обратить внимание, что проверка решений участников на региональном этапе будет выполняться на freepascal, там, в отличие от ABC Pascal, тип
integer является 2-байтовым знаковым и имеет ограничение -32768..32767
longint является 4-байтовым знаковым и имеет ограничение $-2^{31}..2^{31-1}$

Ниже в тексте после каждой задачи приведены краткие объяснения по ее решению. Для лучшего понимания можно просмотреть решения жюри.

Задача 1. Подстрока в числе

Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 мегабайта
Максимальная оценка:	100 баллов
Входной файл	subs_in.txt
Выходной файл	subs_out.txt

В любом числе можно выделить некоторую непрерывную последовательность цифр, которая тоже будет некоторым числом. Требуется написать программу, которая находит максимальное натуральное число **X** такое, что десятичная запись числа $\mathbf{X^2}$ является подстрокой в десятичной записи числа **N**. Если такого числа нет, то вывести ноль.

Формат входных данных (допускается чтение с клавиатуры)

Вводится натуральное число **N**. $1 \le \mathbf{N} \le 1\,000\,000\,000$.

Формат выходных данных (допускается вывод на экран)

Выведите максимальное натуральное число **X** такое, что десятичная запись числа $\mathbf{X^2}$ является подстрокой в десятичной записи числа **N**.

Система оценивания

Баллы начисляются за каждый пройденный тест.

Максимум – 100 баллов.

Примеры входных и выходных данных

№ теста	Входные данные	Выходные данные
1	21	1
2	333	0
3	9646251	25

КОММЕНТАРИИ К РЕШЕНИЮ

Для данной задачи файловый ввод-вывод необязателен.

Данная задача элементарно решается перебором по возможным значениям X от корня из N до 1 с использованием строк и соответствующих функций для их обработки: str и pos. Данное решение приведено.

Однако, участники, не умеющие работать со строками, могут решить задачу с использованием массивов. Хотя, такой вариант требует хорошей техники программирования и является достаточно трудоемким. Это решение тоже приведено жюри. Сложность такого алгоритма линейная, поэтому ограничения по времени не будут превышены.

Задачу можно считать не сложной для понимания, но требующей навыков программирования.

Задача 2. Полином

Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 мегабайта
Максимальная оценка:	100 баллов
Входной файл	poli_in.txt
Выходной файл	poli_out.txt

Любой степенной полином **N**-ой степени может быть задан своими коэффициентами от a_1 до a_{n+1}:

$a_1*X^n + a_2*X^{n-1} + a_3*X^{n-2} + a_4*X^{n-3} +\ldots.+ a_{n-1}*X^2 + a_n*X^1 + a_{n+1}*X^0$

Например, полином 2 степени будет выглядеть так: $a_1*X^2 + a_2*X + a_3$.

Требуется написать программу, которая возводит заданный полином в натуральную степень **K**.

Например, для набора из 4 коэффициентов: **1 –3 0 5** , получим полином 3-й степени и возведем его в квадрат:

$(X^3 – 2X^2 + 5)^2 = X^6 – 6X^5 + 9X^4 + 10X^3 – 30X^2 + 25$.

Результат необходимо вывести тоже в виде коэффициентов от b_1 до b_{n*k+1}. Для указанного примера получаем набор коэффициентов для результата: **1 –6 9 10 –30 25**.

Формат входных данных

Первая строка входного файла содержит два целых числа **N** и **K**. $1 \le N, K \le 8$. Следующие **N**+1 строки содержат коэффициенты входного полинома от $\mathbf{a_1}$ до $\mathbf{a_{n+1}}$, значение каждого из коэффициентов целое, по модулю не превосходит десяти.

Формат выходных данных

В выходной файл необходимо вывести коэффициенты результирующего полинома от $\mathbf{b_1}$ до $\mathbf{b_{n*k+1}}$, разделенные пробелом.

Система оценивания

Правильное решение для $1 \le N, K \le 2$ наберет 30 баллов.

Правильное решение для $1 \le N \le 8$, $1 \le K \le 2$ наберет 30 баллов.

Максимум – 100 баллов.

Баллы начисляются за каждый пройденный тест.

Примеры входных и выходных данных

№ теста	Файл poli_in.txt	Файл poli_out.txt
1	3 2 1 -3 0 5	1 -6 9 10 -30 0 25
2	2 5 2 0 0	32 0 0 0 0 0 0 0 0 0 0

КОММЕНТАРИИ К РЕШЕНИЮ

Для данной задачи обязателен файловый ввод-вывод!

1. Решение данной задачи при ограничениях $1 \le \mathbf{N}, \mathbf{K} \le 2$ является элементарным и требует от участника алгебраического понимания умножения одного алгебраического выражения на другое. При этом не требуется хранить данные в массиве, можно завести необходимое количество переменных. В этом случае можно составить простой алгоритм, не содержащий даже циклов.

2. При ограничениях $1 \le \mathbf{N} \le 8$, $1 \le \mathbf{K} \le 2$ получаем произведение: многочлена ($a_1*X^n + a_2*X^{n-1} + a_3*X^{n-2} + a_4*X^{n-3} + \ldots.. + a_{n-1}*X^2 + a_n*X^1 + a_{n+1}*X^0$) на себя. В итоге нужно перемножить каждый член на каждый. Получим произведения: $a_k*X^{n-k+1} * a_m*X^{n-m+1}$ $(1 \le k, m \le N+1)$ при этом коэффициент запишется при степени X, равной $2*N+2-m-k$. При необходимости коэффициенты при одинаковых степенях X суммируются.

Для удобства реализации можно завести одномерный массив **a** и в элементе массива номер i хранить коэффициент при X^i. Тогда для решения достаточно написать два вложенных цикла: по i, j:

a[i + j]:= **a**[i + j] + **a**[i]***a**[j] $(0 \le i, j \le N)$.

Преобразовать исходные данные в указанный вид не представляет сложности.

3. Рассматривая решение под пунктом 2, участник должен увидеть алгоритм умножения одного многочлена на другой. Аналогично п.2 можно рассмотреть умножение многочлена **a** на **b**, где **a** и **b** - массивы коэффициентов. Тогда результат умножения: **c**[i + j]:= **c**[i + j] + **a**[i]***b**[j].

1) Возьмем **b**=**a**

2) Выполним умножение **c** = **a***b, где **a** и **b** многочлены

3) Возьмем **b**=**c**

4) Повторим шаги 2)-3) K-1 раз и получим нужную степень многочлена.

В решении жюри можно обратить внимание на объявленный массивный тип, который упрощает работу с массивами.

Задача 3. Экспорт апельсинов

Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 мегабайта
Максимальная оценка:	100 баллов
Входной файл	orange_in.txt
Выходной файл	orange_out.txt

Экспорт апельсинов из страны Лимонии осуществляется в бочках, установленных на палубе судна на воздушной подушке. Чтобы не нарушать центровку судна в каждой из **N** бочек должно находиться одинаковое количество апельсинов. Причем в каждой бочке могут находиться апельсины только одного из **M** сортов. Зная число апельсинов каждого сорта на складе, требуется по заданным **N** и **M** определить сколько апельсинов можно уложить в каждую бочку, так, чтобы вывезти за 1 рейс максимальный груз апельсинов.

Формат входных данных

В первой строке входного файла содержатся разделенные пробелом целые числа **N** и **M** ($1 \le \mathbf{M}, \mathbf{N} \le 30\,000$). В следующих **M** строках содержится по одному натуральному числу $\mathbf{A_i}$ ($0 \le \mathbf{A_i} \le 30\,000$; $1 \le i \le \mathbf{M}$), обозначающему количество апельсинов **i**-го сорта на складе.

Формат выходных данных

В выходной файл необходимо вывести одно целое число – количество апельсинов в каждой бочке.

Система оценивания

Правильное решение для $1 \le \mathbf{M}, \mathbf{N} \le 100$, $0 \le \mathbf{A_i} \le 30\,000$ наберет 50 баллов.

Максимум – 100 баллов.

Баллы начисляются за каждый пройденный тест.

Примеры входных и выходных данных

№ теста	Файл orange_in.txt	Файл orange_out.txt
1	5 3 1 1 2	0
2	5 2 1000 5	200
3	3 3 500 300 100	250

КОММЕНТАРИИ К РЕШЕНИЮ

Для данной задачи обязателен файловый ввод-вывод!

Для решения задачи необходимо отметить следующий факт:

если есть некоторое предполагаемое число апельсинов в бочке – F, то можно определить число бочек, которые в принципе можно наполнить из имеющегося числа апельсинов на складе: $Nf = \sum_{1 \le i \le M} (A_i \text{ div } F)$.

1 вариант решения.

Перебор всех вариантов F от $\max_{1 \le i \le M} (A_i)$ до 0. Если Nf для некоторого F получается меньше, чем требуемое N, то перебор останавливается. За ответ принимаем предыдущее значение F = F+1.

Следует учесть, что мы получаем два вложенных цикла: первый для перебора F и второй – для подсчета Nf. Теоретически, максимально возможное число операций = 30 000*30 000. Такой перебор не укладывается в 1 сек. Однако, для M и N в пределах 100, такое решение укладывается в 1 сек.

Данное решение приведено жюри.

2 вариант решения.

С учетом предыдущих выкладок задача решается двоичным поиском по ответу.

1) Пусть Lf = 0, Rf = 30 000. Левая и правая границы поиска.

2) Возьмем за F среднее значение = (Lf + Rf) / 2

3) Проверим, насколько подходит F, вычисляя Nf.

4) Если Nf < N, то можно откинуть левую часть и взять Lf = (Lf + Rf)/2, иначе, откидывает правую часть Rf = (Lf + Rf)/2 и продолжаем поиск с шага 2).

5) Процесс останавливается, когда границы будут на расстоянии 1, останется выбрать правильный ответ.

Данное решение тоже приведено жюри.

Задача 4. Ход коня

Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 мегабайта
Максимальная оценка:	100 баллов
Входной файл	horse_in.txt
Выходной файл	horse_out.txt

На шахматном поле, в клетке с координатами **(X; Y)** ($1 \le$ **X**, **Y** ≤ 8) стоит шахматный конь.

Шахматный конь – фигура, которая может перемещаться за один ход на одну клетку по одной координате и на две клетки по другой координате в пределах шахматной доски.

Изначально сумма очков у игрока равна нулю. Но после каждого хода сумма увеличивается на модуль разности квадратов координат текущей клетки. То есть, если конь на очередном ходе встал на клетку с координатами (2; 6), то сумма очков игрока увеличилась на 32 очка ($|2^2-6^2|$).

Требуется написать программу, которая по начальному расположению коня на доске и количеству ходов коня определит максимальную сумму, которую сможет получить игрок.

Формат входных данных

Первая строка входного файла содержит три натуральных числа X, Y и N, разделенные пробелами. $1 \le$ **X**, **Y** ≤ 8, $1 \le$ **N** ≤ 100)

Формат выходных данных

В выходной файл необходимо вывести одно число – максимальную сумму очков.

Система оценивания

Правильное решение для $1 \le$ **N** ≤ 8 наберет 30 баллов.

Максимум – 100 баллов.

Баллы начисляются за каждый пройденный тест.

Примеры входных и выходных данных

№ теста	Файл horse_in.txt	Файл horse_out.txt
1	8 8 1	13
2	2 8 5	241
3	2 8 4	190

КОММЕНТАРИИ К РЕШЕНИЮ

Для данной задачи обязателен файловый ввод-вывод!

1. Быстрое решение данной задачи – написание рекурсии для перебора в глубину. Однако, данное решение не укладывается в 1 сек. для большого

числа ходов из-за большого количества рекурсивных вызовов, поэтому набирает 30 баллов. Данное решение жюри приводит.

2. Полное решение данной задачи предполагает динамическое программирование. При этом с учетом того, что конь несколько раз может встать на одну и ту же клетку, придется хранить срезы поля для каждого хода, чтобы получить следующий набор вариантов.

Разберем алгоритмически процесс поиска решения для данных 2 8 4.

0 ход

	1	2	3	4	5	6	7	8
1								
2								1
3								
4								
5								
6								
7								
8								

Рассмотрим поле перед выполнением первого хода (0 ход). Поставим 1 в клетку (2,8)

1 ход

	1	2	3	4	5	6	7	8
1						36		
2								
3						28		
4							34	
5								
6								
7								
8								

В таблице, отвечающей за первый ход, поставим в клетки, куда можно сходить конем счет, который получится после этого хода (с учетом 1 в первой клетке).
Теперь каждый следующий шаг может начинаться с непустых клеток.

2 ход

	1	2	3	4	5	6	7	8
1								
2				48				96
3					52		76	
4								
5								
6								
7								
8								

В таблице, отвечающей за второй ход, поставим в клетки, куда можно сходить конем счет, который получится после этого хода.
Смотрим таблицу 1 хода: сходить можно:
из (1,6) с суммой 36
в (2,4) (3,5) (3,7) (2,8).
Проставляем значения.

		из(3,6) с суммой 28 в (1,7) (2,8) (4,8) (5,7) (5,5) (4,4) (2,4) (1,5) Проставляем значения При этом в ячейке (2,8) уже стоит значение 96, значение 88, которое туда нужно поставить – меньше, чем то, что уже есть, то есть в эту ячейку можно попасть с большим числом очков – это выгоднее, поэтому оставляем большее число. Аналогично с ячейкой (2,4).

	1	2	3	4	5	6	7	8
1					52		76	
2				48				96
3					52		76	
4				28				76
5					28		52	
6								
7								
8								

		Действуем далее аналогичным образом для ходов из **(4,7) с суммой 34** в (2,8) (6,8) (6,6) (5,5) (3,5) (2,6) Проставляем значения При этом в ячейке (5,5) уже стоит значение 28, а значение 34, которое туда нужно поставить – больше, чем то, что уже есть, выгоднее поставить большее число. В ячейке (3,5) оставляем большее число.

	1	2	3	4	5	6	7	8
1					52		76	
2				48		66		96
3					52		76	
4				28				76
5					34		52	
6						34		62
7								
8								

3 ход		По аналогии строим матрицы для 3 хода и для 4 хода, основываясь на результатах предыдущих ходов.

	1	2	3	4	5	6	7	8
1		51		81		131		139
2			57		97		121	
3		53		73		123		131
4			59		85		129	
5		49		61		87		115
6			61		63		89	
7				67		75		67
8					73		77	

4 ход		После 4 хода таблица содержит все возможные для получения суммы. Из них необходимо выбрать максимальную. Это и будет правильным ответом. Это 191 в ячейке (2,8) Учтем, что в самом начале в 0 ходе мы поставили 1, которую нужно вычесть. Ответ – 190

	1	2	3	4	5	6	7	8
1	57		105		147		179	
2		81		143		171		191
3	67		97		147		179	
4		85		123		151		171
5	85		101		129		155	
6		99		107		129		157
7	109		113		113		115	
8		127		123		117		89

Реализуя данный алгоритм, получаем полное решение задачи, которое легко укладывается в 1 сек. Оценим сложность алгоритма.

На каждом ходе нужно просмотреть всю матрицу – это 8*8 = 64 операции. Если из каждой ячейки просчитывать ходы – это по 8 раз из каждой ячейки. Имеем 64*8 = 512 операций. Максимальное число ходов – 100, то есть всего может быть не более 512*100 = 51200 операций.