1. Угадайка

(Решение задачи — *Киндер М.И.*)

Пусть X — задуманное число. По условию задачи известно значение $S = ((X + a) \cdot b + b) \cdot a$, из которого найдём $X = \frac{S}{ab} - a - 1$. Поскольку задуманное число X — целое и положительное, необходимо проверить, что S делится нацело на ab (остаток равен 0) и $X > 0$. Если ни выполняется хотя бы одно из этих условий, выводим –1.

2. Часы планеты Зеро

(Решение задачи — *Попов А.А.*)

Количество секунд в сутках на планете Зеро равно abc. Тогда число прошедших суток равно целой части от деления t на abc, а остаток от деления на abc совпадает с количеством секунд, прошедших за последние, ещё незакончившиеся, сутки. Именно это время

$$t1 = t \bmod (abc)$$

будет соответствовать показаниям часов. Количество секунд в одном часе на планете равно bc. Целая часть от деления $t1$ на (bc), то есть $hour = t1 \operatorname{div} (bc)$, — это количество «полных» часов, которое прошло за последние сутки. Значение $t2 = t1 \bmod (bc)$ — это число секунд, прошедших за последний, ещё незакончившийся, час. Значение $min = t2 \operatorname{div} c$ — это число «полных» минут, истёкших за последний незакончившийся час. Значение $sec = t2 \bmod c$ — количество секунд, прошедших за последний, ещё незакончившийся, час. В ответе выводим числа $hour$, min, sec.

3. Без пробелов

(Решение задачи — *Гайфуллин Р.Р.*)

Блок из k-значных p-ичных чисел содержит числа из диапазона $[p^{k-1}; p^k - 1]$, следовательно, его длина $L_k = k \times (p - 1) \times p^{k-1}$. Определим, в каком блоке и на какой позиции находится искомая цифра, последовательно вычитая L_1, L_2, L_3, … из заданного n, пока n не станет отрицательным, и затем, вернувшись в предыдущий блок:

```
k := 1;  stepen := 1;
while n > 0 do begin
    l := k * stepen * (p – 1);
    dec(n, l);
    inc(k);
    stepen := stepen * p;
end;
inc(n, l);  dec(k);
stepen := stepen div p;
```

определим, в каком по счету числе и в каком его разряде находится искомая цифра:

```
nom_chisla := (n + k – 1) div k;
raz := n mod k;
if  raz = 0 then inc(raz, k);
```

Найдем значение числа, содержащего искомую цифру, в 10-ичной системе счисления, а затем и саму цифру:

```
if  k = 1 then chislo_10 := nom_chisla
          else  chislo_10 := stepen + nom_chisla – 1;
for i := 1 to k – raz do
    chislo_10 := chislo_10 div p;
cifra := chislo_10 mod p;
if  cifra < 10 then writeln(figure.out, cifra)
                 else  writeln(figure.out, Chr(Ord('A') + cifra – 10));
```

4. Кошка и Мышка

(Решение задачи — *Киндер М.И.*)

Пусть M_1, M_2, M_3 — точки, соответствующие выходам из норки. Если треугольник $M_1M_2M_3$ — остроугольный, то кошка K должна занять позицию в центре описанного вокруг $M_1M_2M_3$ круга. Если этот треугольник — тупоугольный (или прямоугольный), то положение кошки — в середине наибольшей стороны $M_1M_2M_3$. Если точки M_1, M_2, M_3 лежат на одной прямой, то положение кошки будет по-прежнему в середине отрезка наибольшей длины. Во всех случаях положение кошки определяется однозначно.

5. Трафареты

(Автор задачи и решения — *Киндер М.И.*)

Решение задачи основано на использовании динамического программирования, и основная идея такого решения заключается в следующем. Подсчитаем $m[n]$ — количество способов нарисовать непересекающиеся отрезки с вершинами в заданных n точках.

Выделим одну из точек, например, первую, и разобьем способы на два непересекающихся класса: те, которые содержат отрезок, выходящий из точки 1, и те, в которых такого отрезка нет. Количество способов первого класса, очевидно, равно $m[n-1]$. Все способы второго класса содержат некоторый отрезок, выходящий из точки 1. Пусть, это будет отрезок, соединяющий точки с номерами 1 и $k+1$. По разные стороны от этого отрезка расположено $k-1$ и $n-k-1$ точек. Количество способов провести непересекающиеся отрезки на первой дуге равно $m[k-1]$; количество таких способов на второй дуге равно $m[n-k-1]$. По правилу произведения общее число всех таких комбинаций равно $m[k-1]\cdot m[n-k-1]$. Суммируя по всем k от 1 до $n-1$, получим окончательную формулу для подсчета чисел $m[n]$:

$$m[n] = m[n-1] + \sum_{k=1}^{n-1} m[k-1]\cdot m[n-k-1],$$

с начальным условием $m[0] = 1$.

Приведем несколько первых значений последовательности $m[n]$:

$$1,\ 1,\ 2,\ 4,\ 9,\ 21,\ 51,\ 127,\ \ldots\ .$$

Замечание. Последовательность $m[n]$ играет важную роль в дискретной математике. Эти числа называются *числами Моцкина* (http://oeis.org/A001006), они описывают количество способов провести непрерывную ломаную в верхней полуплоскости, составленную из векторов (1; 1), (1; 0) и (1; –1), начинающуюся в начале координат и заканчивающуюся на оси абсцисс. Приведем явную формулу через биномиальные коэффициенты:

$$m[n] = \sum_{k=0}^{[n/2]} \frac{1}{k+1} C_n^{2k} C_{2k}^{k}.$$

6. 38 Попугаев

(Автор задачи и решения — *Киндер М.И.*)

Пусть попугай $П_1$ выдрал перо у $П_2$, тот — у $П_3$ и так далее. Такая цепочка, в которую, возможно, войдут не все попугаи, замкнётся. Аналогично составим цепочку из части оставшихся попугаев, и так далее до конца. Теперь можно указать, как поступить удаву. Он должен рассаживать попугаев из цепочек поочередно в разные клетки: $П_1$, $П_3$, … — в одну клетку, $П_2$, $П_4$, … — в другую. Если цепочка состоит из *нечётного* числа попугаев, то одного оставшегося попугая придётся проглотить. (При любом способе рассаживания попугаев по клеткам нам не удастся рассадить нечётное число попугаев из этой цепочки по двум клеткам в соответствии с условием задачи.) Если же в цепочке *чётное* число попугаев, то попугаев этой цепочки удаётся рассадить требуемым образом: всех попугаев с нечётными номерами размещаем в одну клетку, с чётными номерами — в другую. Таким образом, наименьшее число попугаев, которое необходимо съесть удаву, совпадает с *количеством нечётных циклов в ориентированном графе «драк»*.

Реализовать предложенный алгоритм можно, например, с помощью обхода графа «в глубину». Сначала всем вершинам графа присвоим метку 0. Затем первую вершина графа пометим

меткой 1 (покрасим её в первый цвет). Вершина графа, связанная с ней ребром, получит метку 2. Смежная с ней вершина снова получит метку 1, и так далее до тех пор, пока не попадём в вершину, которая была уже помечена раньше. В этом случае помеченную вершину пропускаем и переходим к следующей непомеченной вершине графа. В результате этого шага все вершины графа будут покрашены (помечены) в первый или второй цвет.

```
procedure find(i, c: integer);
  begin
          if col[i] <> 0 then exit;
          col[i] := c;
          find(P[i], 3 – c);
   end;
fillchar(col, sizeof(col), 0);
for i := 1 to n do
    if col[i] = 0 then find(i,1);
```

На следующем этапе алгоритма заново просматриваем все вершины графа. Все вершины первого цвета помещаем в клетку 1, все вершины второго цвета — в клетку 2. Если *смежные* вершины оказались одного цвета, то одну из двух вершин удаляем (соответствующего попугая удав съедает), увеличивая при этом счётчик «съеденных» попугаев: inc(eat).

```
num[1] := 0; num[2]:= 0; eat:= 0;
for i := 1 to n do begin
     if col[i] = col[P[i]] then
        begin inc(eat); continue; end;
      inc(num[col[i]]);
      Cage[col[i], num[col[i]]] := i;
end;
```

Председатель жюри М.И.Киндер