Разбор задачи «Кружок английского (только для 7-8 классов!)»

Решение: разбор случаев. Нужно перебрать все возможные варианты разбить учеников на пары и посчитать сколько там «эффективных» пар. Из всех вариантов выбрать оптимальный. Пример кода решения:

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    freopen("english.in","r",stdin);
    freopen("english.out","w",stdout);

    int a, b, c, d, L;
    cin >> a >> b >> c >> d >> L;
    if ((a+b>=L && c+d>=L) || (a+c>=L && b+d>=L) || (a+d>=L && b+c>=L))
        cout << 2;
    else if (a+b>=L || c+d>=L || a+c>=L || b+d>=L || a+d>=L || b+c>=L)
        cout << 1;
    else cout << 0;

    return 0;
}
```

Разбор задачи «Игра со строкой»

Требовалось просто промоделировать то, что описано в условии задачи. Пример решения:

```
#include <iostream>
#include <string>
#include <stdio.h>
using namespace std;

int main()
{
    freopen("abc.in","r",stdin);
    freopen("abc.out","w",stdout);

    int n;
    cin >> n;

    string str;
    for (int i=1; i<=n; i++)
        str = str + "a";
    int ans = 0;
    while(1)
    {
        if (str=="a") break;
        if      (str[0]=='a') str = str + "bc";
        else if (str[0]=='b') str = str + "a";
        else                  str = str + "aaa";
        str = str.substr( 2, str.size()-2 );
        ans++;
```

```
    }
    cout << ans;

    return 0;
}
```

Данная задача — моделирование гипотезы Коллатца при помощи так называемой `2-tag system`. Гипотеза Коллатца — одна из открытых проблем в математике. А именно — никто не знает для любого ли n указанная последовательность строк в конце концов сойдется к «a», хотя для $n \leqslant 100$ это легко проверить.

https://en.wikipedia.org/wiki/Collatz_conjecture
https://en.wikipedia.org/wiki/Tag_system#Example:_Computation_of_Collatz_sequences

Следует отметить, что для некоторых $n \leqslant 100$ количество шагов может быть довольно большим. Например, для $n = 27$ ответ 40656, а для $n = 63$ ответ 42160 (максимум по всем $n \leqslant 100$). Для этих двух n максимальная длина строки в процессе моделирования достигает 4618 (что тоже является максимумом среди всех $n \leqslant 100$). Тем не менее, решение получало 60 баллов при прохождении всех тестов, в которых нужно было смоделировать менее 1000 шагов, а длина сроки в процессе не превышала 250.

Разбор задачи «Генератор кроссвордов»

Нужно было аккуратно реализовать то, что описано в условии задачи. Пример решения:

```
#include <iostream>
#include <string>
#include <stdio.h>
using namespace std;

int n, m;
char T[50][50];   // сжатый кроссворд
char C[300][300]; // разжатый кроссворд

int main()
{
    freopen("crossword.in","r",stdin);
    freopen("crossword.out","w",stdout);

    cin >> n >> m;
    for (int a=1; a<=n; a++)
    {
        string tmp;
        cin >> tmp;
        for (int b=1; b<=m; b++)
            T[a][b] = tmp[b-1];
    }

    // заливаем все "черным" цветом
    for (int a=1; a<=6*n-1; a++)
        for (int b=1; b<=6*m-1; b++)
            C[a][b] = '#';
    // идем по всем клеткам
    for (int a=1; a<=n; a++)
        for (int b=1; b<=m; b++)
            if (T[a][b]=='.') // если клетка "белая"
            {
```

```
                // рисуем рамочку 5x5
                for (int c=a*6-5; c<=a*6-1; c++)
                    for (int d=b*6-5; d<=b*6-1; d++)
                        if (c==a*6-5 || c==a*6-1 || d==b*6-5 || d==b*6-1)
                            C[c][d] = '.';
            }
    // горизонтальные соединения
    for (int a=1; a<=n; a++)
        for (int b=1; b<=m-1; b++)
            if (T[a][b]=='.' && T[a][b+1]=='.')
                for (int c=1; c<=5; c++)
                    C[a*6-3][b*6-3+c] = '.';
    // вертикальные соединения
    for (int a=1; a<=n-1; a++)
        for (int b=1; b<=m; b++)
            if (T[a][b]=='.' && T[a+1][b]=='.')
                for (int c=1; c<=5; c++)
                    C[a*6-3+c][b*6-3] = '.';

    for (int a=1; a<=6*n-1; a++)
    {
        for (int b=1; b<=6*m-1; b++)
            cout << C[a][b];
        cout << "\n";
    }

    return 0;
}
```

Разбор задачи «Хитрая перестановка»

Конструктивное решение. Перестановки вида

$$(6, 1, 7, 2, 8, 3, 9, 4, 10, 5)$$

$$(6, 1, 7, 2, 8, 3, 9, 4, 10, 5, 11)$$

являются ответами. В задаче могут быть другие похожие решения — они тоже засчитываются. Пример решения:

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    freopen("tricky.in","r",stdin);
    freopen("tricky.out","w",stdout);

    int n;
    cin >> n;
    for (int a=1; a<=n; a++)
        if (a%2==1) cout << (n/2 + a/2 + 1) << " ";
        else        cout << a/2 << " ";

    return 0;
}
```

Разбор задачи «Двухъядерный процессор»

Для $n \leqslant 20$ можно решать любым перебором.

Для $n \leqslant 40$ можно решать перебором+бинпоиском (идея meet in the middle), однако это решение сложнее решения на 100 баллов.

Для $n \leqslant 100$ можно решать при помощи классического рюкзака. А именно: заведем булевый массив dp размера суммарного времени выполнения всех программ s. Назначим $dp[0] =$ `true`. Теперь будем перебирать программы и для каждой из них идти по массиву. Если для программы i время выполнения t_i и для индекса массива j значение $dp[j]$ равно `true`, то $dp[j + t_i]$ следует также присвоить значение `true`. При этом при обработке каждой программы индексы массива следует перебирать в обратном порядке, иначе некоторые программы будут учтены более одного раза. В итоге в массиве dp будут отмечены как `true` все возможные времена выполнения всевозможных подмножеств данных программ.

В конце решения следует пройтись по всем индексам i, для которых $dp[i] =$ `true`, вычислить для них значение $\max(i, s - i)$, а затем взять по всем этим значениям минимум.

Сложность решения $O(n^2T)$, где T — максимальное время выполнения одной программы.

Пример решения:

```
#include <iostream>
#include <stdio.h>
using namespace std;

int n;
int p[110];
bool dp[100500];

int main()
{
    freopen("coreduo.in","r",stdin);
    freopen("coreduo.out","w",stdout);

    cin >> n;
    for(int a=1; a<=n; a++)
        cin >> p[a];

    int sum = 0;
    for(int a=1; a<=n; a++)
        sum += p[a];

    dp[0] = true;
    for(int a=1; a<=n; a++)
        for(int b=sum; b>=0; b--)
            if (dp[b])
                dp[b+p[a]] = true;

    int ans = sum;
    for(int a=0; a<=sum; a++)
        if (dp[a])
            ans = min( ans, max( a, sum-a ));
    cout << ans;

    return 0;
}
```

Разбор задачи «Список файлов (только для 9-11 классов!)»

Суть решения: перебираем ответ в порядке возрастания и проверяем его: разбиваем список файлов на интервалы нужной длины, находим на каждом максимум, считаем ширину вывода, сравниваем с шириной терминала. Как только подошло — выводим ответ.

Выполнение данных операций «в лоб» дает $O(n^2)$, что проходит по времени для $n \leqslant 10^4$ и приносит 50 баллов. Такое же решение, если написать его достаточно оптимально (быстрое чтение из файла, использование только 32-битных чисел, отсечения), дает 60 баллов.

Пример решения на 60 баллов:

```
#include <iostream>
#include <stdio.h>
using namespace std;

int n, w;
int A[100500];

int get_max( int i, int j )
{
    int re = 0;
    for(int a=i; a<=j; a++)
re = max( re, A[a] );
    return re;
}

int main()
{
    freopen("list.in","r",stdin);
    freopen("list.out","w",stdout);

    scanf( "%d%d", &n, &w );
    for(int a=1; a<=n; a++)
        scanf( "%d", &A[a] ); // cin может считывать долго 10^5 чисел

    for (int a=1; a<=n; a++)
    {
        int sum = 0;
        for (int b=1; b<=n; b+=a)
        {
            sum += 1 + get_max( b, min( b+a-1, n ) );
            if (sum > w+10) sum = w+10; // защита от переполнения
        }
        sum--;
        if (sum <= w)
        {
            cout << a;
            break;
        }
    }

    return 0;
}
```

Для получения 100 баллов по данной задаче, нужно ускорить вычисление максимума на отрезке при помощи, например, дерева отрезков. Всего операций нахождения максимума

$$n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \cdots + \frac{n}{n} \approx n \log n.$$

(Приведенная выше сумма (если все поделить на n) называется чистичной суммой *гармонического ряда*, подробности можно посмотреть в Википедии). Каждый запрос к дереву отрезков имеет

сложность $O(\log n)$, что дает итоговую сложность $O(n \log^2 n)$.

Пример решения на 100 баллов:

```
#include <iostream>
#include <stdio.h>
using namespace std;

int n, w;
int A[100500];
int T[500500]; // массив для хранения дерева отрезков

void build( int ind, int L, int R ) // построение дерева отрезков
{
    if (L==R) { T[ind] = A[L]; return; }
    int d = (L+R)/2;
    build(ind*2,L,d);
    build(ind*2+1,d+1,R);
    T[ind] = max( T[ind*2], T[ind*2+1] );
}

int get_max( int ind, int i, int j, int L, int R ) // запрос максимума на отрезке [i,j]
{
    if (i==L && j==R) return T[ind];
    int d = (L+R)/2;
    if (j<=d) return get_max( ind*2, i, j, L, d );
    else if (i>=d+1) return get_max( ind*2+1, i, j, d+1, R );
    else return max( get_max( ind*2, i, d, L, d ), get_max( ind*2+1, d+1, j, d+1, R ) );
}

int main()
{
    freopen("list.in","r",stdin);
    freopen("list.out","w",stdout);

    scanf( "%d%d", &n, &w );
    for(int a=1; a<=n; a++)
        scanf( "%d", &A[a] ); // cin может считывать долго 10^5 чисел

    build( 1, 1, n );
    for (int a=1; a<=n; a++)
    {
        int sum = 0;
        for (int b=1; b<=n; b+=a)
        {
            sum += 1 + get_max( 1, b, min( b+a-1, n ), 1, n );
            if (sum > w+10) sum = w+10; // защита от переполнения
        }
        sum--;
        if (sum <= w) { cout << a; break; }
    }

    return 0;
}
```

Еще участники могут попытаться решить задачу, перебирая ответ бинпоиском. Это решение неверно, поскольку ширина вывода от длины интервала растет не монотонно. Тем не менее, оно иногда угадывает ответ и оценивается в 40 баллов.