A. Transsib. See lecture (section 8)

B. Death star 2

To find a maximum of function we need to find a point where a partial derivative equals to zero. This function is quadratic, so its partial derivative is a linear function. Then we have n linear equations from the first criterion and n linear equations from the second criterion. For the next step see the lecture (section 6).

C. Geometrical Dreams

In this problem we need just to build a matrix and run the Gaussian elimination. So your upsolving for this problem is just to find this matrix.

D. Paul’s Salads.

You need to build a matrix for each ingredient and solve it using the Gaussian elimination.

E. Space Poker 2

In this problem we need just to build a matrix and run the Gaussian elimination. So your upsolving for this problem is just to find this matrix.

F. Kirchhoff’s Law

In this problem we need just to build a matrix and run the Gaussian elimination. So your upsolving for this problem is just to find this matrix.

G. Integer-valued Complex Determinant

If you multiply any row of a matrix by k, then the determinant of this matrix will increase by k times. The determinant of diagonal matrix is the multiplication of diagonal elements. So, to find the determinant you need to run the Gaussian elimination and to store the determinant on each step. In this problem we need to use Gaussian integers. You can read about them on wiki: https://en.wikipedia.org/wiki/Gaussian_integer

I. Heroes 2

The topic of this problem is geometry. So I'm not going to tell you the entire solution, I'll say only about Gaussian elimination. For this problem we need to use the Gaussian elimination to get the vector multiplication in 4d. I've already said about that on our lecture today. So now I'll just show the code:

```
vector<int> Gauss(valarray<double> * A)
{
	vector<int> res;
	int start = 0;
	for(int j = 0; start < 3; j++)
	{
		int best = start;
		for(int i = start + 1; i < 3; i++)
		{
			if(fabs(A[i][j]) > fabs(A[best][j]))
			{
				best = i;
			}
		}
		swap(A[start], A[best]);
		if(doubleEqual( A[start][j], 0) )
		{
			continue;
		}
		res.push_back(j);
		double divisor = A[start][j];
		A[start] /= divisor;
		for(int i = 0; i < 3; i++)
		{
			if(i == start) continue;
			double divisor = A[i][j];
			A[i] -= A[start] * divisor;
		}
		start++;
	}
	return res;
}
struct Plane{

	Point p[4];
	Point norm;

	void calcNorm()
	{
		sort(p, p + 4);
		valarray<double> A[7];
		for(int i = 0; i < 3; i++)
		{
			A[i].resize(5);
			for(int j = 0; j < 4; j++)
			{
				A[i][j] = p[i].c[j] - p[3].c[j];
			}
			A[i][4] = 0;
		}
		vector<int> idx = Gauss(A);
		bool use[4];
		for(int i = 0; i < 4; i++)
		{
			use[i] = false;
		}
		for(int i = 0; i < idx.size(); i++)
		{
			use[ idx[i] ] = true;
		}
		for(int i = 0; i < 4; i++)
		{
			if(!use[i] )
			{
				norm.c[i] = 1;
			}
		}
		for(int i = 0; i < idx.size(); i++)
		{
			int cur = idx[i];
			double & res = norm.c[ cur ];
			for(int j = cur + 1; j < 4; j++)
			{
				res -= norm.c[j] * A[i][j];
			}
			res += A[i][4];
		}
		if(sgn(norm % (M - p[0]) ) < 0)
		{
			norm = Point() - norm;
		}
	}
//...}
```