Contents

1 2D Geometry 2

2 3D Geometry 2

3 Combinatorics 2

4 Data Structures 2

5 Graph Theory 2

6 Number Theory 2

7 Search 2

8 Strings 2

Problem	**Tags**
01 A	
02 B	
03 C	
04 D	
05 E	
06 F	
07 G	
08 H	
09 I	
10 J	
11 K	
12 L	
13 M	

Time	**Meeting Description**	**Check**
030	All Problems Read. Write Tags.	
060	Ace Decided. Choose Coder.	
120	Decide & Order Solveable Problems	
150	Status Check	
180	Status Check	
210	Status Check	
240	Status Check	
270	Status Check	

1 2D Geometry

Listing 1 : Primitives

```
typedef complex<double> point;
struct circle {
point c; double r;
circle(point c, double r):c(c),r(r){}
circle(){}
};
double cross(const point &a, const point &b) {
return imag(conj(a)*b);
}
double dot(const point &a, const point &b) {
return real(conj(a)*b);
}
```

Listing 2 : Triangulation

```
for(int i = 1; i < n-1; i++) {
  pt  ai = pts[i] - pts[i-1],
    ib = pts[i+1] - pts[i];
  area += (conj(ai)*ib).imag();
}
```

2 3D Geometry

3 Combinatorics

Listing 3 : Basics

```
// catalan numbers
long long C(int n) {
  return (C(n-1)*2*n*(2*n-1))/(n*(n+1));
  return NCR(2*n, n) - NCR(2*n, n+1);
  return NCR(2*n, n)/(n+1);
}

// derangements
long long D(int n) {
  return n*D(n-1) + pow(-1, n);
  return (n-1)*(D(n-1) + D(n-2));
}

// iterate over all the subsets with no more than m elements
for (int i = 0; i < (1<<n); i=Integer.bitCount(i) < m ? i+1 : (i|(i-1))+1)

// iterate over all the subsets
for (int i=0; i < (1<<n); i++)
    // iterate over all the subsets of the i-th subset
    for(int i2 = i; i2 > 0; i2 = (i2-1) & i)
        // generate the subset induced by i2
```

4 Data Structures

5 Graph Theory

6 Number Theory

Listing 4 : Gaussian Elimination

```
double* GaussianElimination(int N, double **mat) {
  int i, j, k, l; double t;

  for (i = 0; i < N - 1; i++) {
    l = i;
    for (j = i + 1; j < N; j++)
      if (fabs(mat[j][i]) > fabs(mat[l][i]))
        l = j;
    // partial pivot
    for (k = i; k <= N; k++)
    swap(mat[i][k], mat[l][k]);
    for (j = i + 1; j < N; j++)
      for (k = N; k >= i; k--)
        mat[j][k] -= (mat[i][k] * mat[j][i]) / mat[i][i];
  }

  double *res = new double[N];
  for (j = N - 1; j >= 0; j--) {
    for (t = 0.0, k = j + 1; k < N; k++)
    t += mat[j][k] * res[k];
    res[j] = (mat[j][N] - t) / mat[j][j]; // the answer is here
  }
  return res;
}
```

Listing 5 : Tortoise & Hare

```
 // mu = start of cycle, lambda = cycle length
 ii floyd(int x0) {
  int tortoise = f(x0), hare = f(f(x0));
  while(tortoise != hare)
    tortoise = f(tortoise), hare = f(f(hare));
  int mu = 0; hare = x0;
  while(tortoise != hare)
    tortoise = f(tortoise), hare = f(hare), mu++;
  int lambda = 1; hare = f(tortoise);
  while(tortoise != hare)
    hare = f(hare), lambda++;
  return ii(mu, lambda);
 }
```

7 Search

Listing 6 : Ternary Search

```
long double min() {
    long double lo = -1e6, hi = 1e6, res = 3e6;
    while(fabs(lo-hi) > EPS) {
        long double left = (hi-lo)/3 + lo, right = (2*(hi-lo))/3 + lo;
        long double resL = F(left), resR = F(right);
        if(resL < resR)
            hi = right;
        else
            lo = left;
        res = min(res, min(resL, resR));
    }
    return res;
}
```

8 Strings