Volume 1

Home | Up

Do you ever ever wonder why we keep on doing things that we our self consider as 'wrong' or 'sinful' even when we try to do good? Especially if we set a stricter standard of what are considered as 'sinful'. For me, thinking of something 'bad' (e.g. I want to say bad things about other people) even though I have not actually do it is already a sin. With this kind of standard, I doubt that someone can declare that he is 'clean' since he was born. My belief told me that the wages of sin is death, but God, the creator, has given us incredible opportunity that allowed us to be with Him one day. To be continued in volume 2.

Last updated on: 15 October 2007 08:12:38 PM

Comment on this volume: This is the first volume in this online judge. I assume the first 20 problems are test problems, since they are not found in any ACM ICPC contests. There are some regional contest problems at the end of this volume. The difficulty rating of this volume is medium.

No	Problem Name	*	Algorithm
100-117: Source Unknown			
100	The 3n + 1 Problem	2.5	Ad Hoc
101	The Blocks Problem	5.0	Simulation
102	Ecological Bin Packing	2.5	Ad Hoc
103	Stacking Boxes	5.0	Ad Hoc
104	Arbitrage	5.0	Floyd Warshall
105	The Skyline Problem	4.0	Ad Hoc
106	Fermat vs. Pythagoras	6.5	Math
107	The Cat in the Hat	8.0	I didn't solve this problem, look at Ashic's notes
108	Maximum Sum	7.0	DP
109	SCUD Busters	*	Haven't try yet, Computational Geometry
110	Meta-Loopless Sorts	5.5	Ad Hoc
111	History Grading	5.5	DP (LIS)
112	Tree Summing	4.5	Backtracking
113	Power of Cryptography	2.0	Math
114	Simulation Wizardry	4.5	Simulation
115	Climbing Trees	4.5	Ad Hoc
116	Unidirectional TSP	5.5	DP
117	The Postal Worker Rings Once	6.0	Graph
118-125: Duke Internet Programming Contest - 1993			
118	Mutant Flatworld Explorers	4.5	Ad Hoc
119	Greedy Gift Givers	4.5	Ad Hoc
120	Stacks of Flapjacks	5.5	Sorting
121	Pipe Fitters	7.0	Ad Hoc
122	Trees on the level	4.5	Graph
123	Searching Quickly	6.0	Sorting
124	Following Orders	4.5	Backtracking
125	Numbering Paths	*	WA, dunno where is the error...
126-131: Source Unknown			
126	The Errant Physicist	*	Haven't try yet
127	``Accordian'' Patience	*	Haven't try yet
128	Software CRC	7.0	Math
129	Krypton Factor	*	Haven't try yet
130	Roman Roulette	4.5	Simulation
131	The Physic Poker Player	*	Haven't try yet... not familiar with cards
132-139: New Zealand Division 1 - 1990			
132	Bumpy Objects	*	Haven't try yet
133	The Dole Queue	4.5	Simulation
134	Loglan-A Logical Language	*	Haven't try yet
135	No Rectangles	*	Haven't try yet
136	Ugly Numbers	4.0	DP
137	Polygons	*	Haven't try yet

138	Street Numbers	5.5	Math
139	Telephone Tangles	9.9	WA after re-judge, what are changed?
140-163: Source Unknown			
140	Bandwidth	4.5	Backtracking
141	The Spot Game	*	Haven't try yet
142	Mouse Clicks	3.0	Ad Hoc
143	Orchard Trees	9.9	WA, I hate precision errors !!!
144	Student Grants	5.0	Simulation
145	Gondwanaland Telecom	4.5	Ad Hoc
146	ID Codes	1.5	Ad Hoc
147	Dollars	4.5	DP (Counting Change)
148	Anagram Checker	*	Haven't try yet
149	Forests	*	Haven't try yet
150	Double Time	*	Haven't try yet
151	Power Crisis	4.5	Simulation
152	Tree's a Crowd	4.5	Math
153	Permalex	4.5	Ad Hoc
154	Recycling	3.5	Ad Hoc
155	All Squares	3.0	Backtracking
156	Ananagram	4.0	Anagram
157	Route Finding	*	Haven't try yet
158	Calendar	*	Haven't try yet
159	Word Crosses	*	Haven't try yet
160	Factors and Factorials	4.5	Math
161	Traffic Lights	4.0	Ad Hoc
162	Beggar My Neighbour	4.5	Card
163	City Directions	*	Haven't try yet
164-171: South Pacific Regionals - 1991 (2nd link)			
164	String Computer	6.0	DP (Edit Distance)
165	Stamps	6.0	Ad Hoc
166	Making Change	*	Haven't try yet
167	The Sultan's Successors	6.5	Chess
168	Theseus and the Minotaur	*	Haven't try yet
169	Xenosemantics	*	Haven't try yet
170	Clock Patience	4.0	Simulation
171	Car Trialling	*	Haven't try yet
172-178: South Pacific Regionals - 1992 (2nd link)			
172	Calculator Language	*	Haven't try yet
173	Network Wars	*	Haven't try yet
174	Strategy	*	Haven't try yet
175	Keywords	*	Haven't try yet
176	City Navigation	*	Haven't try yet
177	Paper Folding	*	Haven't try yet
178	Shuffling Patience	*	Haven't try yet
179-185: South Pacific Regionals - 1993			
179	Code Breaking	*	Haven't try yet
180	Eeny Meeny	*	Haven't try yet
181	Hearts	*	Haven't try yet
182	Bonus Bonds	*	Haven't try yet
183	Bit Maps	*	Haven't try yet
184	Laser Lines	8.0	Mine is still WA, look at Ashic's notes
185	Roman Numerals	*	Haven't try yet
186-190: Source Unknown			
186	Trip Routing	5.0	Floyd Warshall + Output-related
187	Transaction Processing	4.5	Ad Hoc
188	Perfect Hash	4.0	Ad Hoc
189	Pascal Program Lengths	*	Haven't try yet
190	Circle Through Three Points	5.0	Math
191-199: Southwestern European Regional Contest - 1995			
191	Intersection	6.5	Math (Computational Geometry)

192	Synchronous Design	*	Haven't try yet
193	Graph Coloring	5.5	Graph
194	Triangle	*	Haven't try yet
195	Anagram	5.5	Backtracking
196	Spreadsheet	*	Haven't try yet
197	Cube	*	Haven't try yet
198	Peter's Calculator	*	Haven't try yet
199	Partial Differential Equations	*	Haven't try yet

Total submit-able problems in this volume: 100
Solved problems: 48
Problems in Wrong Answer list from this volume: 11
Unattempted problems: 41
Total hints in this volume: 52

---

100 - The 3n+1 Problem

One of the simplest problem in this online judge. Simply follows the problem description. The only trap is this sentence: "between and including i and j". "between" means, process all numbers between i and j if i<j or between j and i if j>i.

101 - The Blocks Problem

Complex simulation... be meticulous =)

102 - Ecological Bin Packing

Brute force, just read the problem description carefully and figure out those 6 (yes, only 6) possible combinations, and then choose the smallest.

103 - Stacking Boxes (by: Vahid Ghafarpour)

You can make a DAG (Directed Acyclic Graph) from boxes and then run a topological sort to find maximum path.

104 - Arbitrage (by: Vahid Ghafarpour)

You can use matrix cross, with the cross function: A[i,j] = max(A[i][k]*a[k][j]) for k from 1 to n.

105 - The Skyline Problem

First, store all the height in an array of 20000 integers (because maximum coordinates of building is 10000), initially set to everything 0. Then when you read building per building, update this integers value. If your current building is higher than the one stored in this integer, overwrite else ignore it.

Example:

1 10 4
2 5 5
3 15 6

will be processed like this (the array only store 10 elements for clarity, and this is not the exact way I solve the problem, the example below only to show my idea)

0 0 0 0 0 0 0 0 0 0
10 10 10 10 0 0 0 0 0 0 (from element 1 to 4, 10 > 0, overwrite them)
10 10 10 10 5 0 0 0 0 0 (from element 2 to 4, 5 < 10, ignore, for element, 5 > 0, overwrite)
10 10 15 15 15 15 0 0 0 0 (from element 3 to 6, 15 > 10 or 5, overwrite)

The final answer can be derived from this array:
1 10 3 15 6 0

106 - Fermat vs. Phytagoras (by: Varun Kanade)

Any primitive Pythagorean triplet (m,n,p) is of the form
p=x*x+y*y
m=x*x-y*y
n=2*x*y
where x and y are co prime and at least one of x and y is even. Also all other Pythagorean triplets are simply obtained by taking multiples of these. These are well known results from number theory, and should be used to solve this problem.

107 - The Cat in the Hat (by: Ashic Mahtab)

The problem can be solved in two ways: by building a huge tree or by calculation. I prefer the latter.

You are given the height of the initial cat(H) and the total number of workers(x).

Let the number of cats produced from each hat be N and the total number of generations be g.

So, the height and population of some generations will be as follows:

```
Generation |Height |Number of cats for that generation
0 |H |1
1 |H/(N+1) |N
2 |H/((N+1)^2) |N^2
3 |H/((N+1)^3) |N^3
```

Now we have the height of each worker is 1.
So,
```
   H/((N+1)^g)==1;
```
Again the number of workers is x.
So,
```
   N^g=x;
```

Taking logs,
```
  log(H)==g*log(N+1);
  log(x)==g*log(N);
```

Hence,
```
  log(N+1)/log(N)==log(H)/log(x);
```

We try this for values of N=2.0 onwards. Once N is found, we can easily calculate the number of non-workers and the total height of all (including the workers) the cats.

Critical issues:
1. We can't use this method for input like 1,n. If H==1, just output "0 1\n".

2. We have to be careful when x=1. In that case(workers=1), N has to be 1. So, we don't have to find N by the aforesaid method. We still have to find out the output, though.
Tip: If N=1 (which means x=1), H will be a power of 2. Hence, the total number of non-workers will be log(H)/log(2) and the total number of cats will be
log(H)/log(2)+1.

So, 61 1 is valid input and the output is 6 127.

3. I used long doubles for the problem. Precision is very irritating here. You may like to use something like:
```
  temp=x-1;
  if (temp<0)
    temp*=-1;
  if(temp<0.5) {
    /*code*/
  }
```
to check for equality.

And:

```
  rat=logH/logx;
  prevmin=log(2.0L);

  for (N=1.0;;N+=1.0) {
    min=log(N+1) - log(N)*rat;
    if (min<0)
      min*=-1;
    if (min<=prevmin)
      prevmin=min;
    else
      break;
  }
  N-=1.0;
```

to find N.

And:

```
  nonworkers=0;
  for(long double c=0;c<g-0.005;c+=1.0)
    nonworkers+=pow(N,c);
```

to find non-workers. I know it's cumbersome, but using integers or longs got WA. Using long doubles also got WA until I used these techniques. I think this came up as:

```
long double f;
int x=3;
f=x;
```

Sometimes f becomes 3.000000001 and sometimes 2.99999999999. That's why I used only long doubles and coded in the precision stuff.

Hope this helps. 8->

108 - Maximum Sum (by: Ilham Winata Kurnia, Md. Arifuzzaman)

You cannot use O(n^6) algorithm (6 nested loop) to solve this problem. The best solution that I know so far is O(n^3), however, most people use O(n^4) algorithm and still got accepted (within time limit).

Hints by Arif Uzzaman:

Use an array a[100][100] to store data and create another array sum[100][100].

In sum[i][j] you will store sum of values from a[0][0] to a[i][i] like this:

```
for (k=0; k<=i; k++)
  for (l=0; l<=j; l++)
    sum[k][l] += a[k][l];
```

You should do this procedure for each i,j in between the range but you will get TLE.
To avoid TLE you should calculate sum[i][j] using dynamic programming like this:

```
sum[i][j] = sum[i-1][j] + sum[i][j-1] - sum[i-1][j-1] + a[i][j];
```

After fill up two dimensional array sum, check the sum of every rectangle using brute force.
To do this you should use array "sum" now.

If you want to calculate the sum of rectangle from a[i][j] to a[k][l] the rectangle sum is equals sum[k][l]-sum[i-1][l]-sum[k][j-1]+sum[i-1][j-1].

This algorithm does not get TLE.

110 - Meta-Loopless Sorts (by: Vahid Ghafarpour)

Your program should make steps of a sort function, for example bubble sort, it can be very easy to make it with bubble sort.

111 - History Grading

Ignore the background story. Just read the problem description. This is actually a Dynamic Programming problem, a modified Longest Increasing Subsequence (LIS). Click here to see the algorithms.

112 - Tree Summing (with help from: Andras Bizco)

You must determine whether in a binary tree of integers, there exists a root-to-leaf path whose nodes sum to a given integer. This is naturally can be solved using tree recursion, just beware of the common mistakes below

Common Mistake:
1. "0 ()" is false since *"an empty tree has no root-to-leaf paths, any query as to whether a path exists whose sum is a specified integer in an empty tree must be answered negatively"*.
2. "-1 (-1()())" is true because the value can be negative
3. "77 (77(1()())())" is false because even though we have equal value 77=77 in the root, this value is not a full root-to-leaf path.
4. "-77 (-77()())" is true, be careful with input parsing

113 - Power of Cryptography (Notes by: Zachary, Ximo, and Ishtiak)

A simple formula like this is enough for this problem:`exp(log(p)/n))`. Currently I can only got this formula works for Pascal and I'm not sure why it doesn't works for C/C++.

Notes from Zachary Jones:

The problem you were having to get 113 to work in C and C++ has to do with the precision that the online judge uses in C/C++. By using the modifiers in printf or cout.setf(ios::fixed) and cout.setprecision(0), I get AC. I am not sure why this happens, but it does.

Notes from Ximo Planells Lerma:

I don't known your problem with C/C++ but I got AC simply with this code:

```
#include <math.h>
#include <stdio.h>

int main() {
  double n, p, k;
  while (scanf("%lf %lf", &n, &p) == 2) {
     k = exp(log(p)/n);
     printf("%.0lf\n", k);
  }
  return 0;
}
```

Notes from Ishtiak Zaman:

For problem no 113, power of cryptography, better to use`pow(10,(log10(p)/n))` rather than `exp(log(p)/n))`, because the value of log(base 10)(p) is much lesser than log(base e)(p). This one is surely be accepted in C/C++.

114 - Simulation Wizardry (by: Andoko Chandra)

As the problem name suggest, this is a simulation problem. The only thing you should do is just simulate the game and follow the rules written on the problem.

115 - Climbing Trees

Should be easy as long as you read the problem description carefully. You are given a list of (child, parent) pairs, then you are asked about relation of person a and person b. What you need to do is to traverse up (or down, pick only one) the family tree. Note that a is parent of b is the same as b is the child of a.

Suppose you choose traverse up only. Then for each query pair a and b.
Check if parent(a,b) is true or parent(a,intermediates) ... parent (intermediates,b) is true
Output "parent"

Check if parent(b,a) is true or parent(b,intermediates) ... parent (intermediates,a) is true
Output "child"

Traverse up n times from a, check whether b can reach a's n-th ancestor, record this as m.
if n == 0 and m == 0, output "sibling"
if abs(n-m) == 0, output "min(n-m) cousin" <- remember this case when removed 0 times
else, output "min(n-m) cousin removed abs(n-m)"

116 - Unidirectional TSP

Input: An m*n matrix, each cell contains the cost of passing thru that cell.
Output: A path with minimum path-sum from column 0 (leftmost) to column n-1 (rightmost). The path can wraps horizontally.

Let M(i,j) be the minimum value of Unidirectional TSP problem for row i and column j and A[i,j] be the value of the matrix of size m*n. i and j starts from index 0.

Recurrence Relation:

```
;; For first column (col 0), the minimum value is the array value itself.
M[i][0] = A[i][0];

;; For all column j > 0, the minimum value is between one of these three,
;; plus the value of A[i][j].
M[i][j] = A[i][j] + Min(M[(i+m-1)%m][j-1],
                        M[i][j-1],
                        M[(i+1)%m)][j-1]);
;; To output the path, remember the previous row that we choose at current
;; column. Note: this is the trickiest part in problem 116.
```

The formula above may seems confusing, this is just to simplify the code... Note that (j+1)%m will wraps to 0 if j exceeds m-1, and (j+m-1)%m will wraps back to m-1 if it lesser than 0 (try it to make sure).

DP pseudo code:

```
for (i=0; i<m; i++) M[i][0] = A[i][0];

// min3(a,b,c) returns the minimum between a, b, and c
for (j=1; j<n; j++)
  for (i=0; i<m; i++)
    M[i][j] = A[i][j] + min3(M[(i+m-1)%m][j-1],
                             M[i][j-1],
                             M[(i+1)%m][j-1]);

min = M[0][n-1];
for (i=1; i<m; i++)
  if (M[i][n-1] < min) min = M[i][n-1];

output(min);
```

117 - The Postal Worker Rings Once (By: Sohel Hafiz)

If all the edges have even degrees an Euler circuit is possible. Therefore sum of all the edges will give the required answer.

For the other case:
The problem says - The graph will have at most two vertices of odd degree. The trick here is that: a graph can not be drawn which has only one vertex of odd degree (can be proved by Handshaking Theorem). So for this case: find the sum of all the edges and add the shortest path from one vertex (having odd degree) to the other odd degree vertex.

Proof: An Euler Path can be drawn starting from one odd degree vertex and finishing at the other one.

Therefore the answer: Euler path distance + shortest distance between the two vertices.

118 - Mutant Flatworld Explorers

Simple simulation problem, from starting point, either turn left, turn right, or go forward. At the end, report the final destination point plus it's direction.

Common Mistake:
1. Don't forget "Robot scent". If a robot "lost" in position (x,y), it will left a "scent" there so that if another robot "lost", it won't lost in position (x,y) again.
2. Beware with coordinates (remember that lower left corner is at coordinate (0,0)).

119 - Greedy Gift Givers

Basically, what you've to do in this problem is to simulate Give and Receive process. You have to read this problem carefully

Even though the input seems scary (there are characters and numbers mixed), but if you use clever techniques, you will be able to parse them easily.

There will always be N+2 lines for each input blocks, 1st line = N, 2nd = Names, next N lines, description for each person

I use this sample input for explanation:

5	This is the number of person involved
dave laura owen vick amr	This are the person, 5 in total
dave 200 3 laura owen vick	This means Dave spent 200 dollars to give 3 people: Laura, Owen, Vick. Each of them receive 200/3 = 66 (MUST BE INTEGER) so there will be 200-66*3 = 200-198 = 2 dollars left, Dave will retain this
owen 500 1 dave	Owen give 500 dollars to dave only
amr 150 2 vick owen	Amr give 150 dollars to 2 people, Vick and Owen, 150/2 = 75. Because 150-75*2 = 0, Amr will not retain anything from his 150 dollars gift
Laura 0 2 amr vick	Laura give "0" money to 2 person (cliché)
vick 0 0	Vick give NOTHING

After doing that simulation, print out each person's money, simple isn't it?
Just don't forget to retain the money (See "dave 200 3 laura owen vick)

120 - Stacks of Flapjacks

This is a sorting problem but with some "restrictions", you can just simply:
1. Move the biggest pie to the top then flip all to the bottom.
2. Find the second biggest, flip it to the top, then flip to the second place from bottom
3. Repeat this process until everything are sorted.

122 - Trees on the level

Parse the input carefully and use array as data structures to represent this Tree. (see programming section regarding Array based Tree representation). From my observation, the depth of the tree will not exceed 14 level, so an array with 2^14 (16384) elements is enough to get accepted. After you manage to store the input data to an array-based Tree representation, then check the completeness of the Tree. If the Tree is complete, print it directly from the array (again, refer to your algorithm books).

123 - Searching Quickly

No, this problem is not about searching... but a special case of sorting called KWIC-indexing. The hardest part of this problem is actually in storing the data appropriately. You may want to store the string dynamically, because the problem never mention the size of words... you only know that in total, all titles doesn't use up to 10.000 chars. Allocate this dynamically.

Then generate all the possible titles after ignoring the words given. Example: title: "Descent of man", word to ignore: "of", then create "Descent of man" (with keyword descent) and another "Descent of man" (with keyword man).

The remaining part of this problem is then a simple sorting based on the keyword (you can use any sorting algorithm that you like) and print it appropriately (keyword = UPPERCASE, the rest = lowercase).

124 - Following Orders

Sort the variable names first, then simply generate all possible permutation and prune the search tree as soon as it violates the ordering constraint given. The input size is "not so big", so brute force like this will be able to pass the time limit.

130 - Roman Roulette

Brute Force simulation.

133 - The Dole Queue

Similar to 130, another Brute Force simulation.

136 - Ugly Numbers (by: Niaz Morshed Chowdhury)

This problem can be solved using:

1. Dynamic Programming, build a list of ugly numbers bottom up, example:

if we know that '1' is the first ugly number and the only prime factors are 2,3,5, then the next ugly number will be min(2*1,3*1,5*1) which is 2.

when we know that '1' and '2' are the first 2 ugly numbers, the next ugly number will be min(2*2,3*1,5*1) which is 3. Note that factor 2 is now multiplied with '2' whereas the rest are still multiplied with '1'.

2. Brute force, enumerate the numbers one by one incrementally, and check whether their prime factors are only 2,3,5... but you have to wait for a very long time. You can just pre-calculate the 1500'th Ugly Number anyway.

3. or you can do a systematic enumeration. See the explanation for problem 443. You can use the same algorithm for this problem. You just need to make some changes as follow:

1. Loop will be 3 (last loop will be omitted)
2. max = 2000000000 is OK.
3. Take the same array size.
4. Quick Sort will be from 1 to n-1.

138 - Street Numbers

This problem is actually simple. What the problem wants us to do is:

1,2,3,4,5,6,7,8
Your house number = 6
When you walk to the left to the end of the street (always until you encounter 1 - The leftmost house), you sum their numbers.... 5 + 4 + 3 + 2 + 1 = 15. Then you walk to the right and do the same thing, 7+8=15
You found out that both values are the same. Output 6 (Your house number), and 8 (The rightmost house number matching this criteria)

To do this, calculate "Sum of left" and "Sum of right", using Arithmetic Progression formula
Then use precalc (Pre Calculation)

Common Mistake:
1. Time Limit Exceeded... Use precalc
2. Correct algorithm is needed to keep variable from overflow, the 10th line will be around 65 million...

140 - Bandwidth

First, I thought this problem is "difficult" since I can't figure out any good algorithm to do it. But after realizing that maximum nodes is 8..., I see that brute force will be able to solve this problem. Use backtracking to enumerate all possible ordering. 8! is "only" 40.000+, then for each ordering, calculate themaximum distance between connected nodesas explained in problem description.

142 - Mouse Clicks

Straightforward Problem, read the problem description carefully.

143 - Orchard Trees (still WA)

This is a computational geometry problem. Given a triangle, how many points (integer points on the grid) is inside this triangle.

Try to make your algorithm as efficient as possible, i.e. the points outside the bounding box of the triangle obviously outside the triangle...

144 - Student Grants

Brute Force simulation. Be careful with problem statement, READ it over & over again. There are hidden traps inside.

145 - Gondwanaland Telecom

Straightforward Problem, read the problem description carefully.

146 - ID Codes

Do you confused why I put 1.5 as difficulty rating for this problem? Hehe... if you implement it manually, yes, this finding the next permutation will be a bit complicated. But if you use C++ #include <algorithm> and use the next_permutation() function, the solution for this problem will be extremely short !!!, search the internet to study this cool function :).

147 - Dollars

Must use Dynamic Programming. See my programming section - Coin Change. Convert the input (from floating point) to integer by multiplying it by 100. Some precision error problem will occurs here. So rather than doing:

integerAmount = (int) (floatingPointAmount * 100), do this:
integerAmount = (int) (floatingPointAmount * 100 + 0.5)!!!

Then, note that the input will always be multiple of 5 cents, so you can actually divide everything by 5 to save time and space. Instead of storing the original coins + notes values, this values: { 2000,1000,400,200,100,40,20,10,4,2,1 }, will be sufficient :). Don't forget to divide the integerAmount above by 5 too of course...

Note: Coin Changing problem also found in problem 357 (Let Me Count The Ways) and problem 674 (Coin Change). You can solve 3 problems using one similar source code. But now they increase the problem size considerably. Use long long or Big Integer whenever necessary.

151 - Power Crisis

Brute force simulation. This is EXACTLY SIMILAR to problem 440 (Eeny Meeny Moo), only change city number. You can solve 2 problems using one source code (with very minor changes) :-)

152 - Tree's a Crowd

Easy histogram problem. To speed up things, sort the values by x, y, and then z-coordinates first. Then for each point (or tree) i, do a scan through this sorted array from [i.x-10 ... i.x+10], because this is the histogram range that you want (0 to less than 10). This way, you can avoid Time Limit Exceeded even though the number of trees is up to 5.000 because there are not that many trees lies in the range [i.x-10 ... i.x+10] for each tree i :).

154 - Recycling

Brute Force, try it all.

155 - All Squares

Straightforward Recursion, keep dividing the original 1024x1024 square to 4 smaller squares according to the problem (stop until square width is 1x1). If the given (x,y) is inside the small square, increase the "total inside" counter. Finally, output this counter value.

156 - Ananagram

Input all words into an array and remove this word if this word is an anagram to any other word in the array. After that, we have an array of words that are "ananagram". Sort them and output them.

160 - Factors and Factorials

Observation: You only need prime numbers < 100, why?
because in this problem 2<=N<=100, and you need to split this N! into it's prime factor, it's obvious that this factor will never exceed 100.

This is the primes below 100, there are 25 of them.
2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97

Simply loop from 2 to N, split each of this value to prime factors and update the prime factor counter.

161 - Traffic Lights

I set up a Boolean array of 18.000 elements (5 hrs * 60 mins/hrs * 60 secs/min), to flags the seconds which are 'green' based on all traffic lights frequency... There is another way to do it using Longest Common Multiple, but let's just pick the easiest one that works. Oh yeah, you may be interested to check problem number 467 - Synching Signals after solving this problem :).

162 - Beggar My Neighbour

Card simulation, straightforward.

167 - The Sultan's Successors

Standard 8 Queens Problem, refer to your algorithm books, see backtracking section. Just find the maximum score from any possible 8 Queens solutions.

170 - Clock Patience

A card simulation problem. Straightforward... just follow the problem description.

184 - Laser Lines (by: Ashic Mahtab)

Using gradients and floating points will make your work harder than it already is. All you need are integers. Suppose you have three points a, b, c.

```
int pos=a.x*b.y + b.x*c.y + c.x*a.y;
int neg=a.x*c.y + b.x*a.y + c.x*b.y;
if (pos==neg)
/*The points are on the same line*/
```

Create an array of 300+ points. Take in the input. Multi-field qsort them according to their x, then y (if xs are equal). Part 1 done.

Part 2:
Your point class should have 3 integers: x,y,index;
AFTER qsorting, use a loop to set the index of each point according to its place in the array.
So, array={(1,2,0),(2,5,1),(2,9,2)....}

Create a vector<int> check
and a vector<POINT> line

use a loop like:

```
for(i=0;i<n-2;++i)
  for(j=i+1;j<n-1;++j)
    for(k=j+1;k<n;++k) {
      if(point[i],point[j],point[k] are on the same line, call the next function.
    }
```

Next function:
This function should check if the three points are on a previous line. If not push them into the line vector. Also, push the indices into the check vector. If two points are on a previous line but the third isn't, then only push the third point. Use the check vector to keep track. This is confusing, so let's take an example:

say the points are a b c d e f
abdf and ace are collinear.
We have to be careful that output doesn't come out like abdf adf bdf ace.
A line has to be represented only once.

186 - Trip Routing

Max cities = 100, Max highways = 200... Floyd Warshall $O(n^3)$ is capable for solving this problem. If there exist two highways a (length: x) and b (length: y), choose a if x<y, otherwise choose b. This greedy choice is optimal. After that, just do Floyd Warshall and carefully output the result as requested...

187 - Transaction Processing

Straightforward problem.

188 - Perfect Hash

Brute force... first, convert all the words into their respective numbers w ('a' => 1, 'bz' => 90, and so on), then pick the smallest w as initial C. Try if this value C satisfy the formula 1: ((C/w[i]) % n != (C/w[j]) % n) for all i and j....

If no, the pick the next C based on another given formula:
pick the largest value of the following formula, for all i and j which doesn't satisfy the previous formula 1 above: min(((C/w[i])+1)*w[i] , ((C/w[j])+1)*w[j])

Repeat the process until you eventually found the answer...

190 - Circle Through Three Points

Pure mathematic problem. Refer to your mathematic book, and be careful with precision error.

191 - Intersection

If you can somehow avoid the precision error, this problem is "simple". Use sweeping algorithm (see your algorithm book regarding Computational Geometry), and try to avoid line equation formula (because this formula cannot handle line with gradient=0 or gradient=infinite).

193 - Graph Coloring

Find & print maximum black nodes given the restriction that 2 adjacent nodes cannot be all black. Use recursion.

195 - Anagram

You need to systematically generate the permutation. Usebacktracking (recursion).
