V4

What if I told you that in order to be in heaven after death, you need to live a perfectly sinless life... No wrongs, no faults, perfect... I think heaven will be almost empty if that is the way to heaven... I believe God does not like heaven being empty since He has done something to make heaven can be populated by us humans. What is that something?... To be continued in volume 5. See previous story in volume 3.

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

Comment on this volume: I think this is the easiest volume among all existing volumes. If you want to solve many 'easy' Ad Hoc problems, try this volume. Maybe the reason is because this problem consists of many high school contest problems, which should be easily solvable by University students who compete in ACM ICPC.

No	Problem Name	*	Algorithm
400-405: South Central Regionals - 1995 (2nd link)			
400	Unix ls	6.5	Output-related
401	Palindromes	4.5	Ad Hoc
402	M*A*S*H	4.5	Simulation
403	Postscript	6.0	Output-related
404	Radar Scopes	*	Haven't try yet, look at at Minhazul's notes
405	Message Routing	5.0	Simulation
406-411: South Central Regionals - 1996			
406	Prime Cuts	4.5	Math (Prime Number)
407	Gears on a Board	*	Haven't try yet
408	Uniform Generator	4.5	Ad Hoc, Math
409	Excuses, Excuses!	5.0	Ad Hoc
410	Station Balance	4.5	Greedy
411	Centipede Collisions	*	Haven't try yet
412-418: East Central Regionals - 1995 (2nd link)			
412	PI	4.5	Math
413	Up And Down Sequences	4.0	Ad Hoc
414	Machined Surfaces	3.0	Ad Hoc
415	Sunrise	*	Haven't try yet, anyone want to help?
416	LED Test	5.0	Backtracking
417	Word Index	4.5	Ad Hoc
418	Molecules	*	Haven't try yet
419-425: East Central Regionals - 1996 (2nd link)			
419	Matching Meetings	*	Haven't try yet
420	Supercomputer Selection, The Sequel	*	Haven't try yet
421	Polygonal Puzzle	*	Haven't try yet
422	Word Search Wonder	5.5	Graph
423	MPI Maelstrom	4.5	Floyd Warshall
424	Integer Inquiry	4.5	Math (Big Integer)
425	Enigmatic Encryption	*	Haven't try yet
426-430: Southern California Regionals - before 1989			
426	Fifth Bank of Swamp County	*	Haven't try yet
427	Flatland Piano Movers	*	Haven't try yet
428	Swamp County Roofs	*	Haven't try yet
429	Word Transformation	5.0	Graph
430	Swamp County Supervisors	7.0	WA, isn't this should be similar to 435 ??
432-435: Northwestern European Regionals - 1995			

431	Trial of the Millennium	*	Haven't try yet
432	Modern Art	*	Haven't try yet
433	Bank (Not Quite O.C.R.)	4.5	Backtracking
434	Matty's Blocks	6.0	Ad Hoc
435	Block Voting	6.0	Ad Hoc
436-443: University of Ulm Local Contest - 1996			
436	Arbitrage (II)	5.5	Floyd Warshall
437	The Tower of Babylon	4.5	Backtracking
438	The Circumference Of The Circle	3.0	Math
439	Knight Moves	4.5	Graph Traversal
440	Eeny Meeny Moo	3.5	Simulation
441	Lotto	3.0	Ad Hoc
442	Matrix Chain Multiplication	5.5	Ad Hoc
443	Humble Numbers	4.5	DP
444-450: Harding University Local Programming Contest - Fall 1992			
444	Encoder and Decoder	6.5	Ad Hoc
445	Marvelous Mazes	2.0	Ad Hoc
446	Kibbles "n" Bits "n" Bits "n" Bits	5.0	Math (Base Number)
447	Population Explosion	4.0	Simulation
448	OOPS!	4.0	Ad Hoc
449	Majoring in Scales	*	Haven't try yet
450	Little Black Book	4.0	Ad Hoc
451-456: ODU ACM Fall Programming Contest - 1991			
451	Poker Solitaire Evaluator	9.9	WA... dunno where is my mistake...
452	Project Scheduling	6.0	Graph (DAG shortest path)
453	Intersecting Circles	*	Haven't try yet
454	Anagrams	5.0	Anagram
455	Periodic Strings	5.5	Ad Hoc
456	Robotic Stacker	*	Cannot be judged yet!!!
457	Linear Cellular Automata	4.0	Simulation
458	The Decoder	0.5	Ad Hoc
459	Graph Connectivity	5.0	Graph
460	Overlapping Rectangles	5.0	Ad Hoc
461	The Reservation Maker	*	Cannot be judged yet!!!
462	Bridge Hand Evaluator	4.5	Simulation
463	Polynomial Factorization	*	Haven't try yet
464	Sentence/Phrase Generator	5.0	BNF Grammar
465	Overflow	4.0	Math
466	Mirror, Mirror	6.0	Array Manipulation
467	Synching Signals	4.5	Ad Hoc
468	Key To Success	4.0	Ad Hoc
469	Wetlands Of Florida	5.0	Graph (Flood Fill)
470	Nasty Virus	*	Cannot be judged yet!!!
471	Magic Number	5.0	Math
472	Simultaneous Equations	*	Cannot be judged yet!!!
473	Raucous Rockers	*	Haven't try yet
474	Heads / Tails Probability	4.0	Math
475	Wild Thing	*	Cannot be judged yet!!!
476	Points in Figures: Rectangles	3.0	Math (Computational Geometry)
477	Points in Figures: Rectangles and Circles	3.5	Math (Computational Geometry)

478	Points in Figures: Rectangles, Circles, Triangles	5.0	Math (Computational Geometry)
479	Irrigation Flow Rates	*	Cannot be judged yet!!!
480	Tempus Fugit	*	Cannot be judged yet!!!
481	What Goes Up	5.5	DP (LIS)
482	Permutation Array	3.5	Ad Hoc
483	Word Scramble	4.0	Ad Hoc
484	The Department of Redundancy Department	3.0	Ad Hoc
485	Pascal's Triangle of Death	5.0	Ad Hoc + Math (Big Integer)
486	English-Number Translator	5.0	Ad Hoc
487	Boggle Blitz	4.5	Backtracking + Sorting
488	Triangle Wave	2.5	Ad Hoc
489	Hangman Judge	4.0	Ad Hoc
490	Rotating Sentences	4.0	Ad Hoc
491	Tile Topology	*	Haven't try yet
492	Pig Latin	3.5	Ad Hoc
493	Rational Spiral	5.5	Ad Hoc
494	Kindergarten Counting Game	2.0	Ad Hoc
495	Fibonacci Freeze	4.5	Math (Big Integer) + DP
496	Simply Subsets	4.0	Ad Hoc
497	Strategic Defense Initiative	4.5	DP (LIS)
498	Polly The Polynomial	3.5	Math
499	What's The Frequency, Kenneth?	2.0	Ad Hoc

Total submit-able problems in this volume: 100
Solved problems: 57
Problems in Wrong Answer list from this volume:16
Unattempted problems: 27
Total hints in this volume: 71

---

400 - Unix ls

First, for those who don't know. UNIX operating system has a command "ls" <- 'L' + 'S' not 'IS'. This "ls" command is similar to DOS command "dir". What we need to do here is to simulate this command. Must be very careful...

401 - Palindromes

A modified version of standard palindrome... Create a list of characters + its mirror.

Common Mistake:
1. The word 'regular' is now exist (again) in the test case, they have re-judged this problem.
2. Character "S" has "2" as its mirror and vice versa.

402 - M*A*S*H

A simulation problem... but must be very careful...

403 - Postscript

If you have 2-3 hours to spend, then try this problem :-). This problem is not difficult, but the you have to be patient and very very careful !!! It is very straightforward, but you will encounter a lot of small errors, be careful.

404 - Radar Scopes (by: Mohammad Minhazul Alam)

To detect Equipment Warning, one need to calculate the total distance traversed by the plane. If the average is not within 10% of it then Equipment Warning. The other cases are self defined in the problem.

$$d=\sqrt{d1^2 + d2^2 - 2*d1*d2*\cos(angle1-angle2)}$$

The squawk numbers need to be printed with leading zeros (if present in the input).

405 - Message Routing

This problem is a simple but tedious simulation problem. Just do what they want. Be careful with the details...

406 - Prime Cuts

Generate the primes, take the middle one..., simple but need precision.

408 - Uniform Generator (by: Niaz Morshed Chowdhury, notes by: Mohammad Moghimi)

This is a very easy problem but the problem description made it complicated. In this problem you will be given two numbers STEP and MOD. You will have to generate random number using the following formula:
seed(x+1) = [seed(x)+SETP]%MOD

If all the number between 0 and MOD-1 is generate by using the given input then print 'Good Choose' otherwise 'Bad Choose'.

Notes by: Mohammad Moghimi

This problem can be solved mathematically. The solution is that if and only if gcd(step, mod) is 1, they are a good choice.

Solving Technique (Trial and Error)

At first take a 'long' Array (suppose bank) of 100010. Then fill it with 1000001. Now you need to take '0' (Zero) as the initial value of seed. Then generate the numbers and put them in the bank array at the same index. Such as number is N. Then it will be like this, bank[N]=N; After finishing the whole process check the bank array from 0 to MOD-1 for the number 1000001. If you don't get this number in between the given interval then print 'Good Choose' otherwise 'Bad Choose'. Be careful about right-justifying (to avoid Presentation Error).

409 - Excuses, Excuses!

This problem involves complex array handling.

First, you place all the keywords in an array of 20 elements, then you place all the excuses in another array of 20 elements too. (That's the problem specification), then you uppercase all the keywords and excuses then you put it to another array. Why? because It simplifies this problem.

Then you set an array (size 20) of integer, to count how many keywords are there in each excuses. I use a simple method in Pascal -> POS (substr,string) to find out if the keyword is in the excuse.

After that, do a looping to search which one contains the most keyword, and display it.

Common Mistake:
1. Stop your program after finding the excuse that contains the most keyword. Maybe there are 2 or 3 excuses contain same amount of keyword.
2. You must not ignore case. This program is case sensitive, therefore I suggest you to uppercase everything.

410 - Station Balance

Pad the input with zeroes to make them multiple of two, and then sort them. Greedily pair the first with the last, the second with the second last, and so on. This greedy strategy works.

412 - PI (with help from: Yudha Irsandy)

This problem is about finding the approximation of PI from a given data set.

Put the data set into an array, do double looping to find elements with no common factor (using Euclid's GCD algorithm) and total possible combinations

```
for ct1:=0 to n-2 do
  for ct2:=ct1+1 to n-1 do
    begin
      inc(tot);
      if gcd(arr[ct1],arr[ct2])=1 then inc(nocommonfactor);
    end;
```

Use the formula (stated in the problem) : sqrt((6.0/counter)*tot)

Note: 6 in the formula is REALLY 6 !!! "6" in (6/pi^2 ~ 6/10) are DIFFERENT "6", 6/pi^2 is a constant "6" where 6/10 is taken from our NoCommonFactor calculation !!!

413 - Up and Down Sequences

This problem is actually easy. Once you fully understand what the problem want, you can quickly write a program to solve it. This what I did:

1. Store the values to an array (to compare the value with the next item, and so on)
2. You start from I=0 (first array index)
3. If itemI=itemI+1 then increment NEUTRALVALUE
4. If itemI<itemI+1 then increment TOTALUP
5. If itemI>itemI+1 then increment TOTALDOWN
6. If the first deviation is UP -> add NEUTRALVALUE to TOTALUP

7. If the first deviation is DOWN -> add NEUTRALVALUE to TOTALDOWN
8. If there is a deviation change (DOWNWARDS to UPWARDS), inc TOTALUPSEQUENCES
9. If there is a deviation change (UPWARDS to DOWNWARDS), inc TOTALDOWNSEQUENCES
10. Go back to (2) until I>Total array elements
11. Print out the result (Total array elements, TOTALUP / TOTALUPSEQUENCES, TOTALDOWN / TOTALDOWNSEQUENCES)

Common Mistake:
1. Incorrect flagging (You must set flag to UP or DOWN !!!)
2. If TotalUpSequences=0 or TotalDownSequences=0, change their values to 1 or you'll get DIVISION BY ZERO

414 - Machined Surfaces

Straightforward... just simulate the process, count the number of blanks after that.

416 - LED Test

This problem seems confusing at first... but it's just a simulation problem..., try all the possibility, that is, try if you can get a possibly valid 'countdown' sequence. Don't forget that if a segment is burnt, then this segment will still unusable for subsequent checks!!! My suggestion for solving this problem is to generate test cases by yourself, and then test it to your program. If your program can solve it, then you're getting it correct :)

417 - Word Index (by: Ruan David)

1. Make a hash table which contains key a, b, c...z, ab, ac...vwxyz by using 5 for loops!
   It's doable even though it is slow (because there is only 26 alphabets), btw, you actually
   loop 83681 times... Look at the pseudo code below

```
curIndex = 1;

for (i=0; i<26; i++)
  map string ('a'+i) to hashTable with index curIndex
  curIndex++

for (i=0; i<26; i++)
  for (j=i+1; j<26; j++) {
    map string ('a'+i,'a'+j) to hashTable with index curIndex
    curIndex++
  }

// and so on until 5 nested loops... if everything is correct,
// you'll fill in the indexes from 1 to 83681
```

2. Get the input. If it is legal, find it in hash table and output index; otherwise, output zero.

422 - Word Search Wonder (by: Felix Halim)

Backtracking will solve this problem :)

423 - MPI Maelstrom

Find all-pairs-shortest-path (I use Floyd Warshall), and then determine the longest of these minimum connection between processor 1 to any other processor.

424 - Integer Inquiry

Use string to represent this big numbers and do manual addition.

429 - Word Transformation

View this problem as a graph problem.
You have a list of words. These words are the vertices of the graph.
Connect 2 vertices (words) with an edge if they differ in exactly one single letter.
Find the shortest path from starting vertex (starting word) to ending vertex (ending word)
BFS works perfectly here, since BFS can find the shortest vertex from the starting vertex quickly.

433 - Bank (Not Quite O.C.R.)

A backtracking problem. Start with empty digits, add digit from left to right one by one. You need a good data structure (I use array of 10*7, 10 digits, 7 segments/digit) to quickly identify these properties, for example:

```
 _
 _|
 _
```

can be 2,3,8,9, but all with some segments missing... (convince yourself that this is correct), whereas

```
 _
 _|
|_
```

can be exactly 2 (no segment missing), or 8 (with some segments missing)... (convince yourself that this should be the correct one)

Now, backtrack all these combination of possibilities, make sure that at the end of the 9th digits, the sum (according to the formula given) is divisible by 11, and number of digits that must be corrected <= 1. Otherwise print ambiguous or failure, respectively.

435 - Block Voting

Study the definition of power index as described in the problem, and then calculate all the power index for each party.

436 - Arbitrage (II)

Similar to problem 104, use all-pairs-shortest-path algorithm to find whether an arbitrage cycle exist...

437 - The Tower of Babylon (by: Felix Halim)

Find the highest tower that can be built using the unlimited given types of blocks. Even though there are unlimited amount of blocks, we can only use the same type of blocks maximum three times because of the rule that states the upper blocks should have strictly smaller base. As a result, we can simplify the recursion by using blocks without rotating it at all by changing 1 type of block with three same blocks with different base. This will eliminate the needs of rotating the blocks.

438 - The Circumference of the Circle

If you study well during your Senior High School, this problem should be easy.
Note: the following formula is not the only way to compute circumference of the circle!

First, compute a,b,c where a,b,c are the length of Triangle side
Find S where S is (a+b+c)/2 (1/2 of triangle's circumference)
Find L =sqrt(s*(s-a)*(s-b)*(s-c))
Compute r:=(a*b*c)/(4*l) (R= half diameter of the circle)
Output (2*Pi*r) (the circumference of circle)

439 - Knight Moves

You can always find a Knight tour from a square to another square in a chessboard. The problem is, can you find the shortest. Breadth First Search will do.

Common Mistake:
1. Forget to initialize minmoves with maxint (anything big)
2. Wrong recursion, you've to include all 8 knight possible movements

440 - Eeny Meeny Moo

This is just a modification from problem no 151, You only need to change 13 (Wellington at problem no 151) to 2 (Ulm at this problem)

441 - Lotto

If you read the problem carefully, you'll realize that you can solve this problem by using 6 nested loops $O(n^6)$. It is something like this: (j,k,l,m,n,o are variables used for loops, x is the number of elements.)

```
for j:= to x-5 do
  for k:=j+1 to x-4 do
    for l:=k+1 to x-3 do
      for m:=l+1 to x-2 do
        for n:=m+1 to x-1 do
          for o:=n+1 to x do
```

442 - Matrix Chain Multiplication

Read in the matrices, and then calculate the number of multiplication needed to multiple the matrices using the parentheses given.

443 - Humble Numbers (by Niaz Morshed Chowdhury)

According to the problem description we need to find the numbers whose only prime factors are 2,3,5 and 7. So, we can represent the numbers as 2^i x 3^j x 5^k x 7^l. For the different values of i,j,k, and l we will get different numbers. As the maximum range is 5842 and from the last sample input and output we get that 5842nd number is 2000000000. So our maximum value will be this value. Now, by using 4 loops we can generate and preserve
all the humble numbers in between 1 and 2000000000. After that we need to sort them. But here we must use at least O(n log n) sort. Quick Sort is enough for this problem.

Be careful about giving output.
1. After 11,12 and 13 there will be th
2. After 21....91 there will be st

3. After 22....92 there will be nd
4. After 23....93 there will be rd
5 Always after 1,2 and 3 there will be st, nd and rd respectively.

Look at the samples

Input
1
11
1001
100
99

Output
The 1st humble number is 1.
The 11th humble number is 12.
The 1001st humble number is 387072.
The 100th humble number is 450.
The 99th humble number is 448.

Note: This problem can also be solved using Dynamic Programming (refer to problem 136)

444 - Encoder Decoder

Read line by line (this is okay since max chars per line = 80)
If it contains numbers => encoded message, decode the line. (reverse their rule)
If it contains characters => normal message, encode the line. (using their rule)

445 - Marvelous Mazes

You only have to do what this problem wants you to do, i.e:
Read one character per character...
If you encounter 'b', print space
If you encounter '!', print line break (change line)
else, print that character

Be careful with your output, whitespaces and linebreaks

446 - Kibbles n Bits n Bits n Bits (by: Sayutee)

The problem is pretty straightforward. You have to do base conversions only. The input is two Hexadecimal number and the output is two binary numbers and one decimal number. It is like this:

input : Number1(Hex) Operator Number2(Hex)
output: Number1(Bin) Operator Number2(Bin) = Number3(Dec)

For changing to Hex to Bin, simply divide it up with 2 and taking the reminder into a string.
Then just reversed it.

If you use C, you need not change the base from 16 to 10 Explicitly. Just use %X for input and %d for output. The compiler will change that. while changing to binary, use the Hexadecimal number as if it is a decimal the compiler will do whatever needed for you. Use the two statements below:

Input -> scanf("%X %X", &number1, &number2);
Output -> printf("%d", result);

447 - Population Explosion

A problem similar to 457, called 'Game of Life' in Biology. Again, what you can do is to simulate the process and output accordingly. :)

448 - OOPS!

A reverse engineering task. Given a binary compiled code, using the Assembler rule given, reconstruct back the code. A bit tedious I think...

450 - Little Black Book (with help from: Reuber's webpage)

A simple record sorting problem. Simply read the input, tokenize it and put each token into appropriate place in a well defined record, then sort these database by last name, and then output it again using the given format.

The problem is how big is the data? i.e. how many person in the input data?, after several trial and error (yes, I wasted a lot of submissions just to gather this data...), I found out that a line in the input data can be very long, so allocate enough buffer for gets (maybe 1 million chars), and total persons will not exceed 50000... I don't know the exact lower limit, I don't want to waste any more submissions :p

451 - Poker Solitaire Evaluator

This should be just a tedious but straightforward problem... However, I already spent 2-3 hours trying to debug my code, without finding any more error(s) that I can think of... But still WA -_-'''... So, rather than stressing my brain by finding where is the bug... can anyone sent me an idea or some very very difficult test cases for this problem to check my code?

452 - Project Scheduling

This is the first problem that I solve using DAG Shortest Paths algorithms (refer to my programming section or CLRS chapter 24).

The input format is tricky... You have to construct a dependency graph from the given input, and store all weight of vertices somewhere else, then do a Topological Sort (refer to CLRS chapter 23). From this topological order, compute the Critical Path (i.e. find the longest path).

Note that the problem DIDN'T say that ALL vertices will be in a single connected component!!!, make sure your algorithm can cater this, for example:

1

A 1
B 2

Task A (1 day) and Task B (2 days) can both be executed in parallel, so total time needed is 2 days only (and not 3 days or 1 day...)

454 - Anagrams

Max input = 100 lines, maximum O(100^2 ~ 10.000), this is acceptable :)

Just do a brute force matching per every pair of line i and line j. Two lines are anagram if they have the same frequency of characters (ignoring whitespaces). Of course you can do speed-ups by storing their character frequencies first, then do this O(10.000) comparison.

455 - Periodic Strings

A periodic string will always start from the beginning of a string, then that substring will repeat itself until the END of the string

Example:
HoHoHo, “Ho” will repeat itself 3 times (with length 2)
HoHoHoH, “HoHoHoH” will repeat itself only once (with length 7), ie periodic string = original string

Since this problem wants us to find theSMALLEST substring length, we start searching periodic string from the smallest first

My solution will require O(N^2), the outline of my solution is like this:

```
Outer loop (Incrementing the length L of substring, starting from 1) {
  Create a substring with length L
  Inner loop (starting from 1 to length of the original string) {
    check the occurrence of substring, if it occurs from index 1 till the end of
    original string, print current L, then exit program

   if substring does not match the original string, break and go back to outer loop
   to increment L
  }
}
```

The full code to implement this is up to you:-)

Common Mistake

1. MULTIPLE INPUT !!!! (I got crash just because of this)
2. This program states that max length of string is 80, so don’t worry about using array of characters, standard string will be enough.

457 - Linear Cellular Automata

Just simulate this according to their rules. This problem is called a 'game of life' in Biology.

458 - The Decoder

VERY EASY !!!!!!! Just shift each character ASCII Codes 7 characters down. (7 is perfect number :-)

459 - Graph Connectivity

You are to find the maximal connected sub graph in a given graph.

Use Flood Fill (refer to programming section if you don't know flood fill) to find the total connected sub graph. After that count the single nodes that are not part of any sub graph. Add this single nodes to total connected sub graph.

Example:

1

E
AB

Output:

4

Why? because A connected with B (A-B) -> 1 connected sub graph. And you have 3 more free nodes (C,D, & E), 1+3 => 4, output 4

Note: if you know how to use disjoint forest data structure (Union-Find), then you can solve this problem easily by counting how many sets left after you perform all the possible Unions.

460 - Overlapping Rectangles

2 Rectangles are considered overlapping if both of them occupy the same area.

In this problem, you are given 2 rectangles, located in 2 places (it can be the same place) and you've to determine which area those 2 rectangles overlap, or print "No Overlap" if those 2 rectangles not overlap.

To determine whether 2 rectangles overlap or not, the only thing you can do is simulate it

This is what I do:
I need 12 variables to store coordinates
xll1,yll1,xur1,yur1
xll2,yll2,xur2,yur2
overxll,overyll,overxur,overyur

Then simulate all possible combinations of 2 rectangles placement. This is what I found out (I put an ASCII ART to help you)

Rectangle 1 is NOT overlap with Rectangle 2	1--1 2--2 | | | | 1--1 2--2
Rectangle 1 on the left side, Rectangle 2 on the right side, and they overlap	1-21-2 | || | 1-21-2
Rectangle 2 on the left side, Rectangle 1 on the right side, and they overlap	2-12-1 | || | 2-12-1
Rectangle 1 on the upper side, Rectangle 2 on the lower side, and they overlap	1-1 2-2 1-1 2-2
Rectangle 2 on the upper side, Rectangle 1 on the lower side, and they overlap	2-2 1-1 2-2 1-1
Rectangle 1 is INSIDE Rectangle 2, the overlapping region is THE WHOLE Rectangle 1	2---2 |1-1| |1-1| 2---2
Rectangle 2 is INSIDE Rectangle 1, the overlapping region is THE WHOLE Rectangle 2	1---1 |2-2| |2-2| 1---1
Rectangle 1 is PARTIALLY INSIDE Rectangle 2 (there are 4 more possible combinations from this)	1-1 2|-|2 ||-|| |1-1| 2---2
Rectangle 2 is PARTIALLY INSIDE Rectangle 1 (there are 4 more possible combinations from this)	2-2 1|-|1 ||-|| |2-2| 1---1

Common Mistake:
1. MULTIPLE INPUT !!!
2. You must be VERY careful when testing this, try all test case below and make sure your program produce the correct output

462 - Bridge Hand Evaluator

When I first read the problem statement. I can figure out that the solution for this problem will consists of many "if" or "switch" statements, and it's proven to be true. This problem is "easy", but you must make sure you follow all the rules correctly.

464 - Sentence/Phrase Generator

You are given the Backus Naur Form (BNF) format of this language. Then by initializing k = 0 at the start of your program, produce the correct output based on the BNF rule & k. Recursive procedure is naturally fits this problem.

465 - Overflow (with help from: Syukri)

Hehe, I have a trick to solve this problem. Rather than computing anything, use long double, do the calculation as usual (it won't overflow using this data type), then just check whether the values are within the value of *long double maxLongInt = 2.147.483.647.00*

466 - Mirror Mirror

An array manipulation problem. Easy, yes... but you must be very careful. I don't really like array manipulation problem...

467 - Synching Signals

The easiest way to solve this problem is to set a Boolean array of seconds with size is_green[3601], that is, 60 minutes * 60 seconds/minute (because the problem said that max synching time is one hour).

Then, sweep is_green[] array with true for each green region of each signals, and false for yellow/red region for each signals.

Then simply recheck this array, find the first 'true', which is the first second all signals turns green :). convert to minutes and seconds appropriately. Or output 'unable to synch after one hour' if you can't find any true elements in this arrayNote that an output of 60 minutes and 0 seconds shouldbe considered a successful synchronization

468 - Key to Success

Read the input, count their frequencies... sort them... map them... and then decipher it...

I'm not sure about the details, but since my standard solution works, I think there will not be any ambiguity in the test case... in which there won't occur two characters mapped into the same thing..., i.e., they will have different relative frequency.

469 - Wetlands Of Florida

Find how big is the wet area (W). Do DFS Flood Fill from starting coordinate, finding adjacent ‘W’s (8 directions), increment counter if we find adjacent ‘W’, then flag that place as ‘visited’. Note that this problem is in multiple input format and thus making the input reading very complex. Don't forget to restore the graph after each flood fill, because they can ask the same wet coordinate in the next input query...

471 - Magic Numbers

Basically, check them all.
Start from s1 = n, s2 = 1, increase them as follows:

```
s1 += n;
s2 ++;
```

Until s2 == n;

Then check whether s1 and s2 doesn't have repeated digits. Using this pattern, you'll satisfy the s1/s2 = N constraint and always satisfy the property that requires us to sort the output in order of increasing s1 (since s1 will be increased by 'n' every iteration).

476 - Points in Figures: Rectangles

See 478 for explanation.

477 - Points in Figures: Rectangles and Circles

See 478 for explanation.

478 - Points in Figures: Rectangles, Circles, Triangles (with help from: Syukri)

This problem is pure math. Yes, Computer Science is very close to mathematic. Btw,

You can send your Accepted solution for 478 to number 477 and 476, since 478 is the superset of these three problems.

To determine whether a point is in figure, use the following rule:
Note: The problem states that (x,y) will never coincide with a figure border.

For Rectangles:

A point (x,y) is contained in a Rectangle (xUpperLeft,yUpperLeft,xLowerRight,yLowerRight) if and only if (x>xUpperLeft && x<xLowerRight) && (y<yUpperLeft && y>yLowerRight)

For Circles:

A point (x,y) is contained in a Circle (x0,y0,Radius) if and only if
(x-x0)^2 + (y-y0)^2 < Radius^2

For Triangle:

A point (x,y) is contained in a Triangle (x1,y1,x2,y2,x3,y3) if and only if
abs(a1 + a2 + a3 - a4) <= 0.000001

Note for triangle:

a1,a2,a3 are the area of 3 small triangle (with point (x,y) in one of it's side) inside the original triangle.
a4 is the area of the original triangle.
If the point is inside the original triangle, then a1+a2+a3 must beequal to a4
If the point is outside the original triangle, then a1+a2+a3 will belarger than a4
Try it yourself by drawing a simple diagram.
I use epsilon to guard against miscalculation. (epsilon=0.000001)
To find the area, use Matrix theorem to find area (refer to your math books)

481 - What Goes Up

A simple problem description that ask you to implement a very efficient Longest Increasing Subsequence algorithm. In fact the standard O(n^2) DP solution for LIS is not fast enough to beat the time limit. You need to use O(n log k) LIS algorithm. For now, search the internet for this algo... I'll write a summary later

482 - Permutation Array

What the problem wants is:

3 1 2
32.0 54.7 -2

54.7 is the 1st position in array
-2 is the 2nd position in array
32.0 is on the 3rd position in array

Simple isn't it.

483 - Word Scramble (by: Sayutee)

For this kind of problem (which never states the maximum size of the lines), never ever read one whole line then start processing!!! Most likely you won't have enough buffer to do that.

Instead, read character by character, buffer them into a small array, as soon as you encounter a space ' ' or other white spaces, quickly reprint these characters in reverse order, and then flush the buffer. This way, you won't get out of memory error :)

The pseudo code will be something like this:

```
Initially set word to ("") (empty string)

While not end-of-input
  Set c = next character in the input stream
  If c = whitespace or punctuation then
     // word is complete
     reverse word
     output word
  Else
    Add c to word
  End-If

  Output c
end-while
```

484 - The Department of Redundancy Department

This problem can be easily solved using big memory and one linear sweep. Set up an array of tuple <number,occurrence>. Then sweep the input data one by one, everytime you encounter a number, check in your list, if exist, increment the number of occurrence, if no, add new entry to the list.

Finally output this list.

485 - Pascal's Triangle of Death

A simple problem actually, but they make the limit too big... 10^60. Use your BigNumber library (if any) to solve this problem. If you don't have the library, attempt other problems first...

486 - English-Number Translator

You need a parsing technique here... Study the grammar and then write a parser to convert these strings into the appropriate numbers.

487 - Boggle Blitz

This problem is not difficult. Prepare a BST (use C++ STL Map to make life easier), then do a backtracking starting from every position in the table (0,0) to (n,n), and go to all 8 directions whenever the adjacent cell has ASCII value greater than current cell. Then if the length of the word is >= 3, just add it to our BST (this BST is sorted first by string length, and then by lexicographic). Finally, after all backtrackings finished. Output the words in BST using preorder (this will be the sorted output).

488 - Triangle Wave

This judge is really strict for this problem !!!

Input Amplitude and Frequencies
Then do nested loops, the outer loop is for frequencies...
Inside outer loop, do an increasing loop and a decreasing loop to print the Wave.

Remember that this problem is in Multiple Input format!!!, be careful

489 - Hangman Judge

A simulation problem, just do what they want...

490 - Rotating Sentences

In my opinion, the easiest way to solve is to store the sentences in an array, and then re-print them all using the desired orientation :)

492 - Pig Latin

Straightforward problem.... but remember that the line can be very long!!! so do the Pig-latin conversion on-fly.

493 - Rational Spiral

Pre-generate the rational numbers, roughly 1.000.000 such valid numbers. Avoid repetitions or division by zero. Use speed ups whenever possible. GCD algorithm will be very useful here.

494 - Kindergarten Counting Game

Given a line containing multiple words (at least one), you must determine how many words are there in that line.

Just think about the way to tokenize the input, and remember this:
A ``word'' is defined as a consecutive sequence of letters (upper and/or lower case).

495 - Fibonacci Freeze

Use Dynamic Programming solution for Fibonacci Freeze plus Big Integer formula (Big Integer addition)

496 - Simply Subset

Definition of subset:
A is a subset of B if and only if every element of A is in B

Definition of proper subset:
A is a proper subset of B if and only if every element of A is in B but there is at least one element of B that is not in A.

Definition of 2 sets being equal:
Set A equals to Set B if and only if A is a subset of B and B is a subset of A.

Definition of 2 sets are disjoint:
2 sets A and B are disjoint if and only if A intersect B yields an empty set.

This problem want us to determine the relation of 2 set, Set A and Set B.

Create an array to store elements of set A and another array to store elements of set B.
Do an intersection of 2 sets. You can do this by removing common elements from those 2 sets. Use the given definitions to determine the relation.

if Set A empty and Set B empty => A equals B
if Set A empty and Set B NOT empty ==> A is a proper subset of B
if Set A NOT empty and Set B empty ==> B is a proper subset of A
if No Intersection ==> A and B are disjoint
if there is at least 1 intersection, but both sets are not empty ==> I'm confused!

497 - Strategic Defense Initiative

A very similar problem to p231-Testing the Catcher

Difference with p231:
1. 497 is Longest Increasing Subsequence where 231 is Longest Decreasing Subsequence
2. in 497 you have to print the path where in 231 you don't have to do that
3. 497 is multiple Input problem

498 - Polly the Polynomial

This problem is straightforward, just pick the polynomial and the x values, evaluate them one by one, print the result... simple :)

499 - What's The Frequency, Kenneth?

The story of "7 days of creation" is good, isn't it :-) ==> Note, this one is fake..., the actual "7 days of creation" is on the holy Bible (Genesis chapter 1) !!!

Btw, your task in this problem is to count how many occurrences of each alphabet and display the alphabet(s) which have the highest occurrences. Create an array of 52 elements (26 for lower case letters, the other 26 for upper case letters), this array will be used to store how many occurrences each letter has. Scan the input, increment the value of occurrence of the corresponding letter. Output the result...
