

DKC3 2007 COMPUTING PROBLEMS (1)

1. Constellation

(25 pts)

The astronomers of the planet Zolo are constantly observing the sky to find new star constellations. Star constellations are based on starlines of the Force. Whenever a starline of the Force exists between two stars, then these two stars belong to the same constellation. A challenge for the astronomers is that the starlines of the Force sometimes change from time to time, making it difficult to keep their constellation map always up to date.

You have been asked by the astronomers to write a program that computes the number of different constellations for the Daily Constellation Report. A constellation is defined as two or more starlines that contain a common star. The input will consist of a sequence of n ($n \leq 20$) input lines describing a starline of the Force. Each starline is specified by four integers $x_1 y_1 x_2 y_2$ indicating the endpoints (x_1, y_1) and (x_2, y_2) of the starline. The coordinate system ranges from 0 to 999 (including the 0 and 999). There are ten test cases. Each test case is separated by an asterisk (*) on a single line. The output of your program should consist of a single line containing the number of constellations detected in the input.

Input File: C:\DKC3\StarsIn.txt

Output File: C:\DKC3\StarsOut.txt

Examples:

Input: 0 0 0 10
0 10 10 10
10 10 10 0
10 0 0 0
1 1 2 1
2 1 2 2
2 2 1 2
1 2 1 1
1 1 2 2
2 2 3 3
*

Output: 2

Input: 5 8 6 4
5 8 5 15
6 5 5 8
0 0 5 0
9 3 6 12
*

Output: 1

2. Tiribaki Rum Fields**(15 pts)**

Enormous deposits of rum were recently discovered beneath the surface of Tiribaki Islands. Engineers have determined the locations of the rum wells but the transportation of this liquid is not solved yet. The company which owns the rumfield decided to build one main pipeline through entire rumfield from east to west. Each well will be connected to this main pipe by a smaller connection pipeline. You must determine where to build the main pipeline to minimize the amount of material needed for connecting pipelines. Input will consist of the number of rum wells n where $n < 20$ and the coordinates of all rum wells (x, y) where $x < 25$ and $y < 25$. Write a program which finds the y -coordinate of the main pipeline for which the sum of lengths of connection pipelines is the smallest whole integer. Each test case is separated by an asterisk.

Input File: C:\DKC3\RumIn.txt
Output File: C:\DKC3\RumOut.txt

Examples:

Input: 5
 10 1
 15 20
 20 16
 10 7
 10 10
 *

Output: 10

Input: 3
 3 5
 6 9
 13 13
 *

Output: 9

3. Factorials**(5 pts)**

The factorial of an integer n , written $n!$, is the product of all the integers from 1 to n inclusive. For example, $5! = 1 * 2 * 3 * 4 * 5 = 120$. Write a program that accepts a number, n ($1 \leq n \leq 20$), as input and prints the factorial of that number.

Input File: C:\DKC3\FactIn.txt
Output File: C:\DKC3\FactOut.txt

Examples:

Input: 7
 Output: 5040

Input: 9
 Output: 362880

4. Jumping Champion**(10 pts)**

Professor Fermat loves everything related to prime numbers. Remember that a prime is a positive number bigger than one and only divisible by 1 and itself. He is now working on a property of a set of primes called the jumping champion. An integer N is called the "jumping champion" if it is the most frequently occurring difference between consecutive primes.

For example, consider the consecutive primes 2 3 5 7 11. The differences between primes are 1 2 2 4. Therefore, for this set of primes, the jumping champion is exactly 2 (occurring two times). He would really like to know what the corresponding jumping champion is for any set of primes. Could you help him?

Problem: Your task is to write a program that, given a lower and an upper bound, calculate the jumping champion of all the primes numbers that are in the defined limits (the upper and lower bound are considered themselves to be inside the limit). There will be ten test cases. Each test case consists of two integers L and U ($0 \leq L \leq U \leq 1000000$), separated by a single space, which represent the lower and upper limits (respectively) to consider. For each test case, output the jumping champion. If there is no jumping champion, then output the number -1.

Input File: C:\DKC3\JumpIn.txt

Output File: C:\DKC3\JumpOut.txt

Examples:

Input: 2 11

Output: 2

Input: 2 5

Output: -1

5. Skew Binary**(30 pts)**

When a number is expressed in decimal, the k -th digit represents a multiple of 10^k . (Digits are numbered from right to left, where the least significant digit is number 0.)

For example: $81307_{10} = (8 \times 10^4) + (1 \times 10^3) + (3 \times 10^2) + (0 \times 10^1) + (7 \times 10^0) = 81307$

When a number is expressed in binary, the k -th digit represents a multiple of 2^k .

For example: $10011_2 = (1 \times 2^4) + (0 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) = 19$

In skew binary, the k -th digit represents a multiple of $2^{k+1} - 1$. The only possible digits are 0 and 1, except that the least-significant nonzero digit can be a 2.

For example: $10120_{\text{skew}} = 1 \times (2^5 - 1) + 0 \times (2^4 - 1) + 1 \times (2^3 - 1) + 2 \times (2^2 - 1) + 0 \times (2^1 - 1) = 44$

The first 10 numbers in skew binary are 0, 1, 2, 10, 11, 12, 20, 100, 101, and 102.

Input will consist of ten test cases. Each test case contains a nonnegative integer, n , in skew binary. For each number, you must output the decimal equivalent. The decimal value of n will be at most $2^{31} - 1 = 2147483647$.

Input File: C:\DKC3\SkewIn.txt

Output File: C:\DKC3\SkewOut.txt

Examples:

Input: 10120

Output: 44

Input: 100

Output: 7

6. Grammar

(45 pts)

Languages can be defined from context-free grammars. Context-free grammars are typically expressed in Backus-Naur Form (BNF). They consist of productions made of variables (on the left) and terminals (in this case digits). The vertical bar (|) represents OR connections. For example, the variable e below can consist of T or T followed by $*$ followed by D (i.e. $T*D$). Any string in the language can be derived from a parse tree of the grammar by subsequently replacing variables until a terminal is reached. Given the grammar below, build the parse trees from the strings given. Output consists of one level of the tree per line. Siblings are grouped together with operator symbols (+ or *). All output will start with E. There will be ten test cases, one per line.

Grammar:

$E \Rightarrow T \mid E+T$

$T \Rightarrow D \mid T*D$

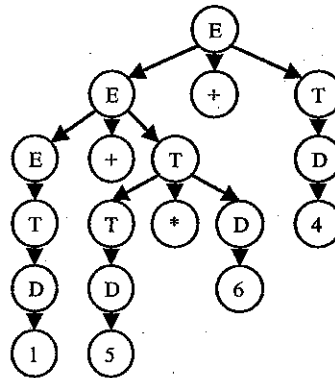
$D \Rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Input File: C:\DKC3\CFGIn.txt

Output File: C:\DKC3\CFGOut.txt

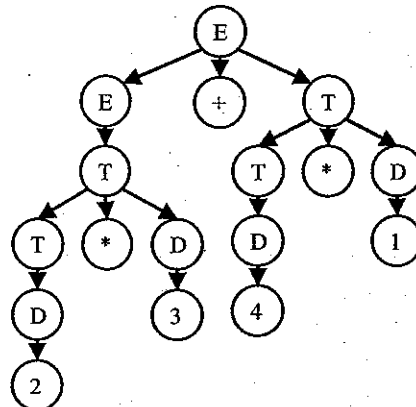
Input: 1 + 5 * 6 + 4

Output:
 E
 E+T
 E+T D
 T T*D 4
 D D 6
 1 5



Input: 2 * 3 + 4 * 1

Output:
 E
 E+T
 T T*D
 T*D D 1
 D 3 4
 2



7. Grid Puzzle

(25 pts)

A logic puzzle is show below. Given a grid of numbers (see upper grid) you are to determine a corresponding grid of letters. Every gray/white cell in the upper grid shows the sum of the numerical values of the letters which appear in the adjoining white/gray cells of the lower grid. So, for example, the 22 in cell A1 is the sum of the letters in B1 and A2 so they must be one of the pairs U and A or T and B or so forth. One letter will not be used. All others will be used only once each. The letters and numbers written outside the 5x5 grids serve no purpose, except to identify cells. 10 test cases will be given each will consist of 5 lines of 5 numbers followed by a coordinate pair and a starting letter at that coordinate. Output for each test case should be the one letter that is not used in the grid.

Letter Values:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

	A	B	C	D	E
1	22	16	47	18	31
2	18	67	22	72	25
3	66	25	87	30	50
4	31	95	29	82	31
5	49	34	66	29	40

	A	B	C	D	E
1					
2			X		
3					
4					
5					

	A	B	C	D	E
1	G	C	H	T	F
2	S	A	X	D	K
3	J	U	I	Q	O
4	Z	E	Y	B	V
5	P	W	M	R	N

Solution to example grid.
'L' is not used.

Input File: C:\DKC3\GridIn.txt
Output File: C:\DKC3\GridOut.txt

Examples:

Input: 22 16 47 18 31
18 67 22 72 25
66 25 87 30 50
31 95 29 82 31
49 34 66 29 40
(C,2) X
*

Output: L

Input: 26 34 58 44 32
29 63 35 53 49
42 29 50 49 31
30 70 50 53 35
41 42 58 37 40
(A,1) O
*

Output: C

8. Abbababa Words**(20 pts)**

The Abbababa Aliens have put out a galactic request for a word verifying machine. Their language consists of only two characters and there is an algorithm that defines all words in their language. However, it can create complex words and sometimes they're unsure of whether a word is in their language or not. They are requesting a program that will take a string and verify it against the language rules to see if it is a legitimate Abbababa word. Abbababa words are defined by the following expression:

$$a^*b^*(ab)^*a$$

where * means 0 or more occurrences of the preceding letter or set of letters – sets are designated by (). If the word is in the language print "Valid". If the word does not match the pattern and is not in the language print the location of the last letter to match the pattern. There will be ten test cases, one test case per line.

Input File: C:\DKC3\AbbaIn.txt
Output File: C:\DKC3\AbbaOut.txt

Examples:

Input: abbababa
 Output: Valid

Input: abaab
 Output: 3

9. Pocket Change**(10 pts)**

Every night when George comes home he splits his change between his two sons, Abraham and Thomas. He always gives each son the same amount, putting any difference in a jar. There will be ten test cases. Each input will consist of a listing of coins (P – penny (1¢) N – nickel (5¢), D – dime (10¢), Q – quarter (25¢)). They will not be in any particular order. Output should consist of three strings, each separated by a space. The first two strings contain the coins given to each son. The third string contains the remaining coins (if any). Note that if the sons will not receive any coins, output should consist of two spaces followed by the remaining coins. Each string should be in reverse numerical order (i.e. Q, D, N, P).

Input File: C:\DKC3\ChangeIn.txt
Output File: C:\DKC3\ChangeOut.txt

Examples:

Input: QDDNPPP
 Output: QP DDNP P

Input: PPPPNQDDQ
 Output: QDNPP QDNPP

10. Counting Out**(15 pts)**

Consider this simple form of shuffling a deck of cards. Hold the deck face down and count to a certain number (say k), on each count taking one card from the top of the deck and placing it at the bottom. When you have finished counting, turn the top card (the $k+1$ 'th card) over and place it face up on the table. Thus cards are continually moved from the top of the deck to the bottom, even when you have fewer cards than your chosen number. Continue in this way until you have no more cards left. Thus if k was 4, you would move 4 cards, one by one, from the top of the deck to the bottom and turn over the fifth. You would then move the sixth, seventh, eighth and ninth cards to the bottom and expose the tenth. Note that the number of the exposed card is always one more than k ; if k was nine you would expose every tenth card, and so on.

Write a program that will read in the chosen number (k) and details of a deck of cards and simulate this process. Remember that a standard deck (or pack) of cards contains 52 cards. These are divided into 4 suits - Spades (S), Hearts (H), Diamonds (D) and Clubs (C). Within each suit there are 13 cards - Ace (A), 2-9, Ten (T), Jack (J), Queen (Q) and King (K).

Input will consist of a series of scenarios. Each scenario will start with a number k ($4 \leq k \leq 26$) on a line by itself. This will be followed by the description of a deck. A deck will be specified on four lines with 13 cards on each line and with cards separated by exactly one space (see the example below). The cards are shown as they would be seen if they were face up, thus the last card in the sequence would be the top of the deck as far as you are concerned. Each test case will be followed by an asterisk (*) on a line by itself. There are ten test cases. Output will consist of a series of lines, one for each scenario in the input. Each line will consist of the last card played, in the format shown below.

Input File: C:\DKC3\CountIn.txt

Output File: C:\DKC3\CountOut.txt

Examples:

Input: 4
 H2 H7 SA HJ D4 S4 HT DJ C9 HQ CT H8 SQ
 DQ S6 D2 H5 CJ CK HA D5 D9 H9 S8 D3 C4
 S2 C5 CQ D8 DT C6 DK H4 CA C8 C2 SK C7
 S5 H6 H3 S9 S7 C3 SJ S3 ST HK DA D6 D7
 *

Output: S8

Input: 10
 H5 D9 SK S4 S8 CQ C4 C5 C2 HA CK S2 D2
 DK CT DT HT D7 C3 S9 SQ S7 SJ HQ D5 SA
 D3 ST C9 H4 CJ C7 D6 HK H8 S3 S5 C8 DJ
 D8 D4 H6 H3 C6 H9 S6 CA DA HJ H2 H7 DQ
 *

Output: SQ

11. Degree Aversion**(5 pts)**

Compute the extent to which an array of eight integers is out of sorted order with sorted order meaning that the smallest index has the smallest value, the second index has the second smallest value, and so on until the largest index has the largest value. The metric for computing the degree of inversion is defined as follows. For each array index i count how many elements at larger indexes have values strictly smaller than the value of the element at index i . The degree of inversion is the sum of this count over all indexes. Notice that if the array is sorted, then the degree of inversion is 0. For each array element x , $1 \leq x \leq 1000$. You can assume that the array size is eight. There will be one test case per line. There are ten test cases.

Input File: C:\DKC3\DegreeIn.txt
Output File: C:\DKC3\DegreeOut.txt

Examples:

Input: 8 7 6 5 4 3 2 1
 Output: 28

Input: 1 2 3 4 5 6 7 8
 Output: 0

12. Veto**(15 pts)**

In the DKC Congress, if a bill is passed with a simple majority by both houses it is then passed on to the President for his signature. A bill becomes law if signed by the President or if not signed within 10 days of being sent to the President and Congress is still in session. If Congress adjourns before the 10 days and the President has not signed the bill then it does not become law ("Pocket Veto.") If the President vetoes the bill it is sent back to Congress with a note listing his/her reasons. The chamber that originated the legislation can attempt to override the veto by a vote of 66 votes in the Senate or 287 in the House, it then goes to the other chamber for an attempted veto override. If the veto of the bill is overridden in both chambers then it becomes law. You will be given a list of the yes and no votes on a bill from each of the houses of Congress. Assuming the President will veto the bill, you must calculate the percentage increase (to the nearest 2 decimal places) in the yes votes in each house of Congress that would be enough to override the Presidential veto. Assume 100 Senators and 435 Representatives are there for the override votes. There will be ten test cases.

Input File: C:\DKC3\VetoIn.txt
Output File: C:\DKC3\VetoOut.txt

Examples:

Input: Senate: Yes:49 No:51
 House: Yes:35 No:400
 *

Output: Senate: 34.69
 House: 720.00

Input: Senate: Yes:62 No:38
 House: Yes:288 No:147
 *

Output: Senate: 6.45
 House: 0.00

13. Election**(20 pts)**

The election of the President of the United States and the Vice President of the United States is indirect. Presidential electors are selected on a state by state basis as determined by the laws of each state. Currently each state uses the popular vote on Election Day to elect electors. Although ballots list the names of the presidential candidates, voters within the 50 states and the District of Columbia are actually choosing Electors from their state when they vote for President and Vice President. These Presidential Electors in turn cast the official (electoral) votes for those two offices. Although the nationwide popular vote is calculated by official and media organizations, it does not determine the winner of the election. The size of the electoral college has been set at 538 since the election of 1964. Each state is allocated as many electors as it has Representatives and Senators in the United States Congress. A candidate must receive a majority of votes from the electoral college (currently 270) to win the Presidency. If no one receives a majority, the election is determined by Congress (the House for presidential candidates, the Senate for vice presidential candidates). The first 51 lines of your input file (shown by A below) will be of the following format: "statename votingpopulation numberofelectoralvotes" This represents the 50 states plus the District of Columbia. Assume that the candidate that wins the majority of the votes in any given area also wins all of the electoral votes from that area. Assume that all possible voters actually vote. Given a list of 51 lines of input per test case (shown by B below) output who won the election (Candidate 1 or 2), and list the number of electoral votes that candidate received, as well as the percentage of the total popular vote that candidate received. Each line of input per test case lists the state and the number of votes received by candidate 1 in that state. There are ten test cases. Each test case is separated by an asterisk.

Input File: C:\DKC3\ElectIn.txt
Output File: C:\DKC3\ElectOut.txt

A:

Alabama 4,599,030 9
 Alaska 670,053 3
 Arizona 6,166,318 10
 Arkansas 2,810,872 6
 California 36,457,549 55
 Colorado 4,753,377 9
 Connecticut 3,504,809 7
 Delaware 853,476 3
 District of Columbia 581,530 3
 Florida 18,089,888 27
 Georgia 9,363,941 15
 Hawaii 1,285,498 4
 Idaho 1,466,465 4
 Illinois 12,831,970 21
 Indiana 6,313,520 11
 Iowa 2,982,085 7
 Kansas 2,764,075 6
 Kentucky 4,206,074 8
 Louisiana 4,287,768 9
 Maine 1,321,574 4
 Maryland 5,615,727 10
 Massachusetts 6,437,193 12
 Michigan 10,095,643 17
 Minnesota 5,167,101 10
 Mississippi 2,910,540 6
 Missouri 5,842,713 11
 Montana 944,632 3
 Nebraska 1,768,331 5
 Nevada 2,495,529 5
 New Hampshire 1,314,895 4
 New Jersey 8,724,560 15
 New Mexico 1,954,599 5
 New York 19,306,183 31
 North Carolina 8,856,505 15
 North Dakota 635,867 3
 Ohio 11,478,006 20
 Oklahoma 3,579,212 7
 Oregon 3,700,758 7

DKC³ 2007 Computing Problems (1)

Pennsylvania 12,440,621 21
Rhode Island 1,067,610 4
South Carolina 4,321,249 8
South Dakota 781,919 3
Tennessee 6,038,803 11
Texas 23,507,783 34
Utah 2,550,063 5
Vermont 623,908 3
Virginia 7,642,884 13
Washington 6,395,798 11
West Virginia 1,818,470 5
Wisconsin 5,556,506 10
Wyoming 515,004 3

B:

Examples:

Input:

Alabama 4,599,030
Alaska 670
Arizona 6,166
Arkansas 2,810,872
California 36,457,549
Colorado 4,753,377
Connecticut 3,504
Delaware 853,478
District of Columbia 581,530
Florida 18,089
Georgia 9,363,941
Hawaii 1,285,498
Idaho 1,466,465
Illinois 12,831,979
Indiana 6,313,520
Iowa 2,982,085
Kansas 2,764
Kentucky 4,206,074
Louisiana 4,287,868
Maine 1,321,574
Maryland 5,615,727
Massachusetts 6,437,193
Michigan 10,095
Minnesota 5,167
Mississippi 2,910,540
Missouri 5,842,713
Montana 944,632
Nebraska 1,768,301
Nevada 2,495,529
New Hampshire 1,314,895
New Jersey 8,724,560
New Mexico 1,954
New York 19,306,183
North Carolina 8,856,505
North Dakota 635
Ohio 11,478
Oklahoma 3,579
Oregon 3,700
Pennsylvania 12,440
Rhode Island 1,067,610
South Carolina 4,321,249
South Dakota 781
Tennessee 6,038,803
Texas 23,507,783
Utah 2,550
Vermont 623
Virginia 7,642
Washington 6,395
West Virginia 1,818
Wisconsin 5,556
Wyoming 515

Output:

Candidate 1: 342 64.59%

DKC³ 2007 Computing Problems (1)

Input:

Alabama 599,030
Alaska 670,053
Arizona 166,318
Arkansas 2,810,872
California 1
Colorado 753,377
Connecticut 504,809
Delaware 853,476
District of Columbia 581,530
Florida 1
Georgia 9,363,941
Hawaii 1,285,498
Idaho 1,466,465
Illinois 12,831,970
Indiana 6,313,520
Iowa 982,085
Kansas 764,075
Kentucky 206,074
Louisiana 287,768
Maine 1,321,574
Maryland 5,615,727
Massachusetts 477,193
Michigan 1
Minnesota 167,101
Mississippi 2,910,540
Missouri 5,842,713
Montana 944,632
Nebraska 1,768,331
Nevada 495,529
New Hampshire 1,314,895
New Jersey 724,560
New Mexico 1,954,599
New York 1
North Carolina 856,505
North Dakota 635,867
Ohio 1
Oklahoma 3,579,212
Oregon 700,758
Pennsylvania 1
Rhode Island 1,067,610
South Carolina 321,249
South Dakota 781,919
Tennessee 038,803
Texas 1
Utah 2,550,063
Vermont 623,908
Virginia 642,884
Washington 395,798
West Virginia 1,818,470
Wisconsin 556,506
Wyoming 515,004

Output:

Candidate 2: 387 73.59%

14. Octospider Time**(20 pts)**

Octospiders (an alien race with 8 limbs) have a different time keeping system than us humans. Their number system is octal instead of decimal and is split up into these units of time:

1 Octospider day = 8 terts
 1 tert = 8 wodens
 1 woden = 8 fengs
 1 feng = 8 nillets

A nillet is a little over 28 seconds long, which makes the Octospider day 32 hours, 14 minutes, and 6 seconds long. Write a program that takes two Octospider times and calculates the difference between them in terts, wodens, fengs, and nillets. Input times will be in the form of an octal number (e.g. 1234 = 1 tert, 2 wodens, 3 fengs, and 4 nillets). The two times for each test case will be separated by a space. There will be one test case per line. There will be ten test cases.

Input File: C:\DKC3\OctoIn.txt
Output File: C:\DKC3\OctoOut.txt

Examples:

Input: 4052 1733
 Output: 2117

Input: 0521 6324
 Output: 5603

15. Lumber Waste**(30 pts)**

Dennis is building his house and wants to reduce the amount of lumber needed, so he has decided to build a geodesic dome instead of a standard rectangular house. Geodesic dome frameworks are made up of different length struts (in this case made out of 2x dimensional lumber) attached to central hubs. Standard lumber lengths at Dennis's local lumber yard are 8', 10', 12', 14', and 16'. Help Dennis minimize his wasted lumber by calculating the length and number of boards he needs to buy to cut out all of his struts while generating the least amount of waste (fewest boards too small to make up another strut). Where there are multiple solutions, choose shorter boards (four 8' boards are cheaper than two 16' boards). Each test case will contain one to six pairs of numbers (separated by spaces) that represent each strut length (in inches) and quantity. Output the quantity of each length of board and the resulting waste (in inches). There will be one test case per line. There will be ten test cases.

Input File: C:\DKC3\LumberIn.txt
Output File: C:\DKC3\LumberOut.txt

Examples:

Input: 89 35 79 30
 Output: 5 8' + 30 14' w/ 35" of waste

Input: 67 30 77 40 79 50
 Output: 30 12' + 30 14' w/ 320" of waste