LABEXAM-PRAC-01_MIDSEM

The D List (p1v1d1)

The D List [25 marks]

Problem Statement
We will give you a list of digits given as **non-negative integers** all separated by a space. The list will be ended by a -1 which is not supposed to be considered a part of the list. You have to process this list and output the following **on different lines**

1. The number of elements in the list (not including the last -1)
2. The number of times the digit 2 appears in the list
3. The number of times the first digit of the list appears in the list
4. If the number formed by the list is divisible by 11, print "YES" (without quotes), else print "NO" without quotes

HINTS: To test the divisibility of a number by 11, take an alternating sum of the digits of the number. For example, if the number is 1234, an alternating sum would be 1 - 2 + 3 - 4. If this sum is divisible by 11, so is the original number. If this sum is not divisible by 11, neither is the original number.

EXAMPLE:
INPUT
3 2 -1

OUTPUT:
2
1
1
NO

Explanation: The list has two digits, the digit 2 appears once, the first digit 3 also appears just once, and the number 32 is not divisible by 11.

Problem-specific Words of Caution:

1. The numbers 0, -22, 110, -44 etc are all considered divisible by 11. However, then numbers -4, 10, 16 etc are not divisible by 11. In particular, note that negative numbers can be divisible by 11 as well.
2. We promise that the first digit will never be zero, as well as that the number formed by the digits will be strictly positive
3. The number formed by the digits we give you may not fit inside int or long variables.

General Words of Caution

1. **Do not forget to submit your code**. You can submit multiple times. Your last submission will get graded.
2. Marks will be allotted for the following

    1. Proper and meaningful variable names
    2. Nice looking and consistent indentation
    3. At least a couple of comments explaining to the human grader what are you doing, especially when the calculations are not obvious
    4. Comments, good indentation and meaningful variable names are very important for the human grader to understand what are you doing and why. If they cannot understand your code, do not expect them to give you (partial) marks either.

3. Solutions that indulge in hard-coding **will get a straight zero** even if they are passing some test cases. Hard-coding is a form of cheating strategy where someone write code of the form "if(input == A) printf(B)" without doing any calculations on A to obtain B. The values of A and B are either read from the evaluation/submission window or else guessed.
4. Questions will be graded by the **autograder as well as a human grader**
5. Be careful about extra/missing lines and extra/missing spaces if you do not want to lose autograder marks
6. Proportion of marks allotted to autograder (in particular, weightage to visible and hidden test cases) and human grader will be revealed when marks and grading rubrics are released
7. The total marks of this exam are 150.
8. You are allowed to use the libraries math.h and stdlib.h **but not any other library**. Use of unpermitted libraries will carry a penalty. You may use programming tools such as arrays, functions, recursion, pointers, in case you are familiar with the use of these. However, you will be given no special credit for using these advanced programming techniques, nor will you receive any help should you face difficulties in using them, for example, TLE or segmentation fault errors. Use these advanced techniques at your own risk.

Grading Scheme:
Total marks: **[25 Points]**

There will be partial grading in this question. There are four lines in your output. Printing each line correctly, in the correct order, carries some weightage. The first two lines are worth 12.5% each. The third line is worth 25% and the fourth line is worth 50% weightage. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
3 2 -1	2 1 1 NO
4 1 0 0 1 3 5 0 4 9 8 5 3 5 2 4 0 5 7 1 1 6 9 7 7 2 3 4 6 5 3 6 -1	32 2 4 YES
7 9 2 9 2 8 2 8 1 3 6 7 3 0 8 1 8 9 4 0 8 9 8 1 9 3 0 3 4 8 0 4 8 7 7 3 5 2 9 1 5 3 6 7 7 2 7 4 0 5 0 3 5	93

2 4 6 3 7 3 7 6 5 1 9 3 9 8 6 9 1 5 4 5 0 3 4 8 1 1 7 9 9 8 4 0 8 1 3 1 7 1 7 4 -1	6 12 YES
9 0 1 8 1 4 6 9 2 3 6 3 0 5 4 1 7 -1	17 1 2 NO
2 2 2 2 2 2 0 0 2 2 0 0 0 0 2 2 0 0 0 0 0 0 0 0 2 2 0 0 0 0 0 0 0 0 0 0 2 2 2 2 -1	40 16 16 YES
1 2 6 2 2 8 1 1 3 8 3 9 6 4 9 1 4 1 0 0 8 6 6 9 9 3 7 5 6 1 0 8 2 8 2 1 3 9 9 2 0 9 8 9 2 5 0 5 0 0 9 6 7 3 4 7 0 6 9 3 5 2 2 0 8 8 6 2 6 4 8 8 6 6 8 1 9 6 9 3 6 4 8 7 0 4 3 6 9 0 0 9 6 5 8 4 7 3 1 -1	99 10 9 NO

The D Factor (p1v2d1)

The D Factors [50 marks]

Problem Statement
You will be given a **strictly positive integer** n. In **two separate lines** print the following

1. On the first line print the distinct divisors of n **in increasing order** with two divisors separated by a space. Divisors include prime and non-prime divisors as well as 1 and the number itself. **There should be no trailing spaces at the end of this line**
2. On the second line, print the smallest strictly positive number M that has **exactly n distinct divisors** (including prime and non-prime divisors and including 1 and the number M itself)

Problem-specific Words of Caution:

1. For the sake of this question, divisors of a number such as M or n, include prime and non-prime divisors, as well as include the number 1 and the number itself.
2. We assure you that all outputs will fit inside integer variables.

EXAMPLE:
INPUT
6

OUTPUT:
1 2 3 6
12

Explanation: The distinct factors of 6 (not necessarily prime and including 1 and 6 itself) are 1 2 3 and 6. The number 12 has 6 distinct divisors (not necessarily prime and including 1 and 12 itself) 1 2 3 4 6 12.

General Words of Caution

1. **Do not forget to submit your code**. You can submit multiple times. Your last submission will get graded.
2. Marks will be allotted for the following
    1. Proper and meaningful variable names
    2. Nice looking and consistent indentation
    3. At least a couple of comments explaining to the human grader what are you doing, especially when the calculations are not obvious
    4. Comments, good indentation and meaningful variable names are very important for the human grader to understand what are you doing and why. If they cannot understand your code, do not expect them to give you (partial) marks either.
3. Solutions that indulge in hard-coding **will get a straight zero** even if they are passing some test cases. Hard-coding is a form of cheating strategy where someone write code of the form "if(input == A) printf(B)" without doing any calculations on A to obtain B. The values of A and B are either read from the evaluation/submission window or else guessed.
4. Questions will be graded by the **autograder as well as a human grader**
5. Be careful about extra/missing lines and extra/missing spaces if you do not want to lose autograder marks
6. Proportion of marks allotted to autograder (in particular, weightage to visible and hidden test cases) and human grader will be revealed when marks and grading rubrics are released
7. The total marks of this exam are 150.
8. You are allowed to use the libraries math.h and stdlib.h **but not any other library**. Use of unpermitted libraries will carry a penalty. You may use programming tools such as arrays, functions, recursion, pointers, in case you are familiar with the use of these. However, you will be given no special credit for using these advanced programming techniques, nor will you receive any help should you face difficulties in using them, for example, TLE or segmentation fault errors. Use these advanced techniques at your own risk.

Grading Scheme:
Total marks: **[50 Points]**

There will be partial grading in this question. There are two lines in your output. Printing each line correctly, in the correct order, carries some weightage. The first line has 25% weightage and the second line has 75% weightage. There are 3 visible and 6 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
2	1 2 2
6	1 2 3 6 12
8	1 2 4 8

	24
15	1 3 5 15 144
1	1 1
20	1 2 4 5 10 20 240
18	1 2 3 6 9 18 180
16	1 2 4 8 16 120
27	1 3 9 27 900

All Charged Up (p1v3d1)

All Charged Up [75 marks]

Problem Statement
Electrostatic charges (either +1 or -1) have been placed at integer coordinates in a 3D space. The x, y, z, coordinates of these charges can all range from 1 to 256 (both limits included). The charge at a given location (i,j,k) is determined by the following rules

1. Case 1: if the location is at the boundary of this arrangement (i.e. outer surface of this 256 x 256 x 256 sized 3D cube), the charge at that location is +1
2. Case 2: if the location is not at the boundary but the sum i + j + k is prime, then also the charge at that location is +1
3. Case 3: if the location is neither at the boundary, nor is the sum of coordinates prime, but still the coordinates (i,j,k) satisfy the property that the quadratic equation i * x^2 + j * x + k has only real roots, even then the location has charge +1
4. Case 4: any location that does not have charge +1 has charge -1

We will give you two locations (a,b,c) and (p,q,r) with all coordinates in the range [1, 256] (both ends included). Consider the **smallest cuboid C that contains both these locations**. Your task is to print the following in **five separate lines**.

1. The number of locations inside or on the boundary of the cuboid C that satisfy case 1
2. The number of locations inside or on the boundary of the cuboid C that satisfy case 2
3. The number of locations inside or on the boundary of the cuboid C that satisfy case 3
4. The number of locations inside or on the boundary of the cuboid C that satisfy case 4
5. The sum of all charges at locations inside or on the boundary of the cuboid C

Problem-specific Words of Caution:

1. The smallest cuboid C that contains the locations (a,b,c) and (p,q,r) may have one or more side lengths zero e.g. the cuboid may be just a flat rectangle.
2. The coordinates of the two points given to you are not necessarily in any particular order. Keep this in mind.

INPUT:
(a,b,c)(p,q,r)

OUTPUT:
case1
case2
case3
case4
totalCharge

EXAMPLE:
INPUT
(1,1,1)(2,2,2)
OUTPUT:
7
0
0
1
6

Explanation: All locations in the cuboid C except the location (2,2,2) lie on the boundary and have +1 charge. There are 7 such locations. The location (2,2,2) is neither on the boundary nor is the sum prime nor does the quadratic equation x^2 + x + 1 have real roots. Hence the location (2,2,2) has -1 charge. Total charge in the cuboid is 7-1 = 6.

General Words of Caution

1. **Do not forget to submit your code**. You can submit multiple times. Your last submission will get graded.
2. Marks will be allotted for the following
    1. Proper and meaningful variable names
    2. Nice looking and consistent indentation
    3. At least a couple of comments explaining to the human grader what are you doing, especially when the calculations are not obvious
    4. Comments, good indentation and meaningful variable names are very important for the human grader to understand what are you doing and why. If they cannot understand your code, do not expect them to give you (partial) marks either.
3. Solutions that indulge in hard-coding **will get a straight zero** even if they are passing some test cases. Hard-coding is a form of cheating strategy where someone write code of the form "if(input == A) printf(B)" without doing any calculations on A to obtain B. The values of A and B are either read from the evaluation/submission window or else guessed.
4. Questions will be graded by the **autograder as well as a human grader**
5. Be careful about extra/missing lines and extra/missing spaces if you do not want to lose autograder marks
6. Proportion of marks allotted to autograder (in particular, weightage to visible and hidden test cases) and human grader will be revealed when marks and grading rubrics are released

7. The total marks of this exam are 150.
8. You are allowed to use the libraries math.h and stdlib.h **but not any other library**. Use of unpermitted libraries will carry a penalty. You may use programming tools such as arrays, functions, recursion, pointers, in case you are familiar with the use of these. However, you will be given no special credit for using these advanced programming techniques, nor will you receive any help should you face difficulties in using them, for example, TLE or segmentation fault errors. Use these advanced techniques at your own risk.

Grading Scheme:
Total marks: **[75 Points]**

There will be partial grading in this question. There are five lines in your output. Printing each line correctly, in the correct order, carries some weightage. The five lines carry 30%, 30%, 10%, 10% and 20% weightage respectively. There are 4 visible and 8 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
(1,1,1)(2,2,2)	7 0 0 1 6
(1,1,1)(1,2,2)	4 0 0 0 4
(1,1,1)(1,1,2)	2 0 0 0 2
(1,1,1)(1,1,1)	1 0 0 0 1
(5,1,1)(5,1,1)	1 0 0 0 1
(2,3,2)(2,3,2)	0 1 0 0 1

(2,4,2)(2,4,2)	0 0 1 0 1
(2,4,8)(2,4,8)	0 0 0 1 -1
(2,2,2)(1,1,1)	7 0 0 1 6
(20,2,20)(1,10,1)	351 830 76 2343 -1086
(3,30,3)(10,1,10)	64 488 877 491 938
(20,20,20)(10,10,10)	0 327 1 1003 -675

The S Factor (p2v1d1)

The S Factor [25 marks]

Problem Statement
You will be given a **long integer** n as input. Using this integer, you have to print the following on **four different lines**

1. The smallest divisor of n that is strictly greater than 1 (notice that this divisor has to be prime)
2. The number of times the digit 3 appears when we write down n
3. The number of times the first digit of n (when seen from the left) appears when we write down n
4. The power of the smallest divisor of n in its prime factorization

Problem-specific Words of Caution:

1. We promise that when we give you the long integer, the first digit of the integer will never be zero
2. We promise that the number we give you will itself never be 0, it will be strictly positive
3. Your outputs may not fit inside int variables - take precautions.

EXAMPLE:
INPUT
123456

OUTPUT:
2
1
1
6

Explanation: The smallest non unity divisor of 123456 is 2. The digit 3 appears once in 123456. The first digit of 123456 is 1 and that also appears only once in 123456. The number factorizes as 64 * 1929 and hence, the power of 2, the smallest divisor, is 6 since 64 = 2^6.

General Words of Caution

1. **Do not forget to submit your code**. You can submit multiple times. Your last submission will get graded.
2. Marks will be allotted for the following

    1. Proper and meaningful variable names
    2. Nice looking and consistent indentation
    3. At least a couple of comments explaining to the human grader what are you doing, especially when the calculations are not obvious
    4. Comments, good indentation and meaningful variable names are very important for the human grader to understand what are you doing and why. If they cannot understand your code, do not expect them to give you (partial) marks either.

3. Solutions that indulge in hard-coding **will get a straight zero** even if they are passing some test cases. Hard-coding is a form of cheating strategy where someone write code of the form "if(input == A) printf(B)" without doing any calculations on A to obtain B. The values of A and B are either read from the evaluation/submission window or else guessed.
4. Questions will be graded by the **autograder as well as a human grader**
5. Be careful about extra/missing lines and extra/missing spaces if you do not want to lose autograder marks
6. Proportion of marks allotted to autograder (in particular, weightage to visible and hidden test cases) and human grader will be revealed when marks and grading rubrics are released
7. The total marks of this exam are 150.
8. You are allowed to use the libraries math.h and stdlib.h **but not any other library**. Use of unpermitted libraries will carry a penalty. You may use programming tools such as arrays, functions, recursion, pointers, in case you are familiar with the use of these. However, you will be given no special credit for using these advanced programming techniques, nor will you receive any help should you face difficulties in using them, for example, TLE or segmentation fault errors. Use these advanced techniques at your own risk.

Grading Scheme:
Total marks: **[25 Points]**

There will be partial grading in this question. There are four lines in your output. Printing each line correctly, in the correct order, carries some weightage. The first two lines are worth 12.5% each. The third line is worth

25% and the fourth line is worth 50% weightage. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
123456	2 1 1 6
135	3 1 1 3
1024	2 0 1 10
333	3 3 3 2
10000	2 0 1 4
190333	11 3 1 4

The S List (p2v2d1)

The S List [50 marks]

Problem Statement
We will give you a **strictly positive integer** n. Given this, you have to print your output **in two separate lines**

1. In the first line, print a list of all prime numbers strictly less than the square root of n. Two primes should be separated by a space. **There should be no trailing spaces at the end of this line**. If there are no such primes, print "NO PRIMES" (without quotes).
2. In the second line print the sum of the values of nCk for all k that are primes strictly less than the square root of n. If there are no such primes, print 0.

$$\left(\begin{array}{c} n\\k\end{array} \right) = \frac{n!}{k!\cdot(n-k)!}$$

Problem-specific Words of Caution:

1. Take care that combinatorial expressions involve factorials that can take huge values that can even exceed the long limit.
2. Even though your input will fit inside an integer variable, your output may require long variables.

EXAMPLE:
INPUT
7

OUTPUT:
2
21

Explanation: sqrt(7) = 2.64. Thus the only prime less than sqrt(7) is 2 and 7 choose 2 is 21.

General Words of Caution

1. **Do not forget to submit your code**. You can submit multiple times. Your last submission will get graded.
2. Marks will be allotted for the following

    1. Proper and meaningful variable names
    2. Nice looking and consistent indentation
    3. At least a couple of comments explaining to the human grader what are you doing, especially when the calculations are not obvious
    4. Comments, good indentation and meaningful variable names are very important for the human grader to understand what are you doing and why. If they cannot understand your code, do not expect them to give you (partial) marks either.

3. Solutions that indulge in hard-coding **will get a straight zero** even if they are passing some test cases. Hard-coding is a form of cheating strategy where someone write code of the form "if(input == A) printf(B)" without doing any calculations on A to obtain B. The values of A and B are either read from the evaluation/submission window or else guessed.
4. Questions will be graded by the **autograder as well as a human grader**
5. Be careful about extra/missing lines and extra/missing spaces if you do not want to lose autograder marks
6. Proportion of marks allotted to autograder (in particular, weightage to visible and hidden test cases) and human grader will be revealed when marks and grading rubrics are released
7. The total marks of this exam are 150.
8. You are allowed to use the libraries math.h and stdlib.h **but not any other library**. Use of unpermitted libraries will carry a penalty. You may use programming tools such as arrays, functions, recursion, pointers, in case you are familiar with the use of these. However, you will be given no special credit for using these advanced programming techniques, nor will you receive any help should you face difficulties in using them, for example, TLE or segmentation fault errors. Use these advanced techniques at your own risk.

Grading Scheme:

Total marks: **[50 Points]**

There will be partial grading in this question. There are two lines in your output. Printing each line correctly, in the correct order, carries some weightage. The first line has 25% weightage and the second line has 75% weightage. There are 3 visible and 6 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
7	2 21
10	2 3 165
75	2 3 5 7 2002159540
49	2 3 5 1926484
100	2 3 5 7 16083014970
25	2 3 2600
26	2 3 5 68705
1	NO PRIMES 0
4	NO PRIMES 0

Smith Numbers (p2v3d1)

Smith Numbers [75 marks]

Problem Statement
Smith numbers are composite numbers for which the sum of the digits of the number is equal to the sum of the digits of all its prime factors (including repetitions).

For example, 378 = 2 x 3 x 3 x 3 x 7 is a Smith number since 3 + 7 + 8 = 2 + 3 + 3 + 3 + 7. Another nice example of a Smith number is 22 = 2 x 11 and is a Smith number since 2 + 2 = 2 + 1 + 1.

You will be given a **strictly positive integer** n. You have to print "Smith Number" (without quotes) in case the number is a Smith number else print "Not Smith Number" (without quotes) in case the number is not a Smith number.

Problem-specific Words of Caution:

1. Take care of spelling and other mistakes
2. As mentioned in the problem statement above, only composite numbers can be Smith numbers, i.e. prime numbers cannot be Smith numbers.

General Words of Caution

1. **Do not forget to submit your code**. You can submit multiple times. Your last submission will get graded.
2. Marks will be allotted for the following
    1. Proper and meaningful variable names
    2. Nice looking and consistent indentation
    3. At least a couple of comments explaining to the human grader what are you doing, especially when the calculations are not obvious
    4. Comments, good indentation and meaningful variable names are very important for the human grader to understand what are you doing and why. If they cannot understand your code, do not expect them to give you (partial) marks either.
3. Solutions that indulge in hard-coding **will get a straight zero** even if they are passing some test cases. Hard-coding is a form of cheating strategy where someone write code of the form "if(input == A) printf(B)" without doing any calculations on A to obtain B. The values of A and B are either read from the evaluation/submission window or else guessed.
4. Questions will be graded by the **autograder as well as a human grader**
5. Be careful about extra/missing lines and extra/missing spaces if you do not want to lose autograder marks
6. Proportion of marks allotted to autograder (in particular, weightage to visible and hidden test cases) and human grader will be revealed when marks and grading rubrics are released
7. The total marks of this exam are 150.
8. You are allowed to use the libraries math.h and stdlib.h **but not any other library**. Use of unpermitted libraries will carry a penalty. You may use programming tools such as arrays, functions, recursion, pointers, in case you are familiar with the use of these. However, you will be given no special credit for using these advanced programming techniques, nor will you receive any help should you face difficulties in using them, for example, TLE or segmentation fault errors. Use these advanced techniques at your own risk.

Grading Scheme:
Total marks: **[75 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 points. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). There are 4 visible and 8 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
22	Smith Number
378	Smith Number
12	Not Smith Number
42	Not Smith Number

627	Smith Number
11	Not Smith Number
1499	Not Smith Number
1000	Not Smith Number
121	Smith Number
94	Smith Number
355	Smith Number
128	Not Smith Number