LAB-PRAC-02_SCAN-PRINT

Mr C goes on a diet (p1v1d1)

Mr C goes on a diet [20 marks]

Problem Statement
Some students told Mr C that he is getting a bit overweight so he decided to go on a diet. To do so, he wants to find the total surface area and volume of the glass he uses to measure his food. His glass is a cylinder with radius r and height h. Help him find our these two quantities.

Your code should take 2 integers as input, the first is radius, the second is height, and then calculate and output the total surface area and volume of the cylinder in two separate lines. Don't forget the high-school formulae for the total surface area and volume of a cylinder :)

Caution

1. You need only output the integral part of the actual output i.e. if the actual volume is 100.56, output 100 and if the actual total surface area is 98.04, output 98.
2. For this question, you may take the value of pi as 22/7.
3. Do not use any datatype other than int.
4. Do not use any library other than stdio.h

HINTS:

1. Visible test case number 1 is there to confirm if you are giving output in the correct format or not. Be careful about extra spaces and lines.
2. Visible test case number 2 is there to check if you are performing integer arithmetic operations properly or not.

INPUT:
radius height

OUTPUT:
area
volume

EXAMPLE:
INPUT
1 2
OUTPUT:
15
6

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. In each test case, 50% marks are for giving the correct total surface area and 50% marks will be for giving the correct volume i.e. if a test case is worth 2 points, 1 point is for correct total surface area and 1 point for correct volume.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if both answers are correct. Thus, if your surface area is correct but volume is incorrect, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get 50% partial marks.

Each visible test case is worth 2 points and each hidden test case is worth 4 points.

All Test Cases (Visible + Hidden)

Input	Output
7 1	198 154
1 3	22 9
1 1	9 3
0 0	0 0
4 0	50 0
0 5	0 0

Permute Password (p1v2d1)

Permute Password [20 marks]

Problem Statement
You forgot the CVV of your ATM card which has 3 digits. Trying to recall it, you were able to remember the digits individually, but not their order. For example, you recall that the digits 5, 8, and 0 were there in the number but you have forgotten if the CVV number is 508 or 805 or something else.

Mr C has offered to help you recover your CVV code. Mr. C will take 3 digits from you and print out all permutations of these digits, one permutation in each line. Your code should take 3 integers as input, each representing one digit of the CVV code. The permutations should be printed in increasing

Caution

1. Permutations should be output in increasing order, one on each line.
2. Do not use any datatype other than int.
3. Do not use any library other than stdio.h

You may assume that

1. The three digits given to you are distinct
2. The digits in the CVV code are also distinct, i.e. no digit repeats in your CVV code
3. You will be given the digits in increasing order, i.e. in the previous example, the digits will be given to you in the order 0, 5, 8.

HINTS: Visible tests are there to show you how to give the output as well as warn you if you have extra spaces or extra lines etc in your output. There should be only 6 lines in your output, one corresponding to each permutation.

INPUT:
digit1 digit2 digit3

OUTPUT:
permutation1
permutation2
permutation3
permutation4
permutation5
permutation6

EXAMPLE:
INPUT
1 2 3
OUTPUT:
123
132
213
231
312
321

Grading Scheme:
Total marks: **[20 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 marks. Please be careful of the order of the permutation (take help of visible test cases) as well as extra spaces and lines. Each visible test case is worth 2 points and each hidden test case is worth 4 points.

All Test Cases (Visible + Hidden)

Input	Output
1 2 3	123 132 213 231 312 321
0 3 4	034

	043 304 340 403 430
2 4 6	246 264 426 462 624 642
2 5 8	258 285 528 582 825 852
1 3 8	138 183 318 381 813 831
0 1 2	012 021 102 120 201 210

Escapes around Tutors (p1v3d1)

Escapes around Tutors [10 marks]

Problem Statement:
Your tutors work very hard to prepare for your tutorials, make questions for your lab assignments, quizzes, and exams. Motivate them by shouting out to them in following manner.

"Tutors"
\Ka Tempo/
%HIGH hai!!!1!!!%

INPUT:
There is no input in this question.

OUTPUT:
"Tutors"
\Ka Tempo/
%HIGH hai!!!1!!!%

Grading Scheme
Total marks **[10 Points]**
There will be no partial grading in this question. A complete match will get full marks whereas incomplete matches will get zero marks. Be careful about extra/missing spaces.

All Test Cases (Visible + Hidden)

Input	Output
	"Tutors" \Ka Tempo/ %HIGH hai!!!1!!!%

Amusing Fractions (p2v1d1)

Amusing Fractions [20 marks]

Problem Statement
Letâ€™s start the day with some fraction arithmetic. You will be given two fractions as input. You are required to add and subtract the two fractions and display the integral part of both these values. For example, if the fractions are -10/2 and -3/4 then we have -10/2 + (-3/4) = -5.75 and -10/2 - (-3/4) = -4.25 and so your answers should be -5 and -4. Be careful -- as these examples demonstrate, the integral part is not found by either rounding up or down - the integral part is found by simply ignoring the fractional part of the number.

Caution

1. Do not use math.h or any header file other than stdio.h
2. Use only integer variables. No floats, doubles etc.
3. Fractions need not be provided to you in their lowest form i.e. the numerator and denominator may have common factors other than one.
4. The fractions we give may be negative, or else their sum or difference may be negative
5. We will never give any of the denominators to be zero so do not worry about divide-by-zero errors

HINTS:

1. Visible test case number 1 is there to confirm if you are giving output in the correct format or not. Be careful about extra spaces and lines.
2. Visible test case number 2 is there to check if you are performing integer arithmetic operations properly or not.

INPUT:
numerator1 denominator1 numerator2 denominator2
OUTPUT:

sum
difference

EXAMPLE:
INPUT
-10 2 3 -4
OUTPUT:
-5
-4

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. In each test case, 50% marks are for giving the correct sum and 50% marks will be for giving the correct difference i.e. if a test case is worth 2 points, 1 point is for correct sum and 1 point for correct difference.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if both answers are correct. Thus, if your sum is correct but difference is incorrect, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get 50% partial marks.

Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible test cases and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
-10 2 3 -4	-5 -4
10 7 33 9	5 -2
-75 3 55 8	-18 -31
-15 2 55 6	1 -16
6 3 2 4	2 1
9 3 5 6	3 2

P and C (p2v2d1)

P and C [20 marks]

Problem Statement
You are given 3 NON-ZERO numbers between 1-9 i.e. they are SINGLE DIGIT. The digits will be given in increasing order. For example, you may be given the digits 3,6,8 in that order. Your job is to print out all positive numbers strictly less than 100 (i.e. 100 not included) that can be created out of these 3 digits. Print

all these numbers in increasing order and one number per line.

Caution

1. The three digits given to you are distinct and are given in increasing order.
2. The numbers you form out of these digits may have repetitions. For example, if you are given the digits 3,6,8, then the number 33 is perfectly valid since it is less than 100 and is formed out of the digits given to you.
3. Your numbers should be output in increasing order, one on each line.
4. Do not use any datatype other than int.
5. Do not use any library other than stdio.h

HINTS: Visible test cases are there to show you how to give the output as well as warn you if you have extra spaces or extra lines etc in your output.

INPUT:
digit1 digit2 digit3
OUTPUT:
number1
number2
...

EXAMPLE:
INPUT
1 2 3
OUTPUT:
1
2
3
11
12
13
21
22
23
31
32
33

Grading Scheme:
Total marks: **[20 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 marks. Please be careful of the order of the permutation (take help of visible test cases) as well as extra spaces and lines. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible test cases and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output

1 2 3	1 2 3 11 12 13 21 22 23 31 32 33
4 6 9	4 6 9 44 46 49 64 66 69 94 96 99
1 6 8	1 6 8 11 16 18 61 66 68 81 86 88
2 4 8	2 4 8 22 24 28 42 44 48 82 84 88
3 4 5	3 4 5 33 34 35 43 44

	45 53 54 55
1 2 7	1 2 7 11 12 17 21 22 27 71 72 77

Build a Rhombus (p2v3d1)

Build a Rhombus [10 marks]

Problem Statement
You will be given a single digit number as input. You need to print the number in the format given below. For example, if the digit is 1, print the following pattern,

```
1
11
111
1111
1111
111
11
1
```

Caution

1. Take good care not to have extra/missing spaces or extra/missing lines in your output. These will fail an exact match and the autograder will give you a zero.
2. Do not use any library other than stdio.h
3. The code to take in an input (using scanf) has already been provided to you.

HINTS:

1. Make sure your code can handle all digits.

INPUT:
A single digit from 0-9

OUTPUT:
A rhombus design made completely out of that digit (see below)

EXAMPLE:
INPUT
1
OUTPUT:
1
11
111
1111
1111
111
11
1

Grading Scheme:
Total marks: **[10 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 marks. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 2 points and each hidden test case is worth 4 points.

All Test Cases (Visible + Hidden)

Input	Output
4	4 44 444 4444 4444 444 44 4
8	8 88 888 8888 8888 888 88 8
0	0 00 000 0000 0000 000 00 0

Developing Interest at IITK (p3v1d1)

Developing Interest at IITK [20 marks]

Problem Statement
Your batchmate applied for a loan to pay for their studies at IITK. The bank calculates loans in an interesting manner. For the first two years, it applies simple interest to the loan amount and for the next two years it applies compound interest on the loan amount. Assume that your batchmate does not pay any loan amount in the 4 years they are spending at IITK so the loan only keeps building up. Also, the interest rate stays the same for all four years.

For example, if the principal loan amount is Rs 100 and annual interest rate is 1%, then the interest calculation would take place as follows

1. First year: starting loan amount Rs 100, simple interest Re 1, final loan amount Rs 101
2. Second year: starting loan amount Rs 101, simple interest Re 1 (recall that simple interest is calculated on principal amount i.e. Rs 100, and not current loan amount), final loan amount Rs 102
3. Third year: starting loan amount Rs 102, compound interest Rs 1% of Rs 102 (recall that compound interest is calculated on current loan amount, and not principal loan amount), final loan amount Rs 103.02
4. Fourth year: starting loan amount Rs 103.02, compound interest Rs 1% of 103.02, final loan amount Rs 104.0502

Your job is to output the integer part of the loan amount at the end of the 2nd and 4th year. In the above example, these amounts are 102 and 104 respectively. Please note that these are not rounded up or rounded down values. If the true amount is 114.79, you should output 114. If the true amount is 78.23, you should output 78.

Caution

1. Do not use math.h or any header file other than stdio.h
2. Use only integer variables. No floats, doubles etc.
3. Principal and rate of interest will be non-negative integers.
4. Do not use loops or conditional statements.

HINTS:

1. Visible test case number 1 is there to confirm if you are giving output in the correct format or not. Be careful about extra spaces and lines.
2. Visible test case number 2 is there to check if you are performing integer arithmetic operations properly or not.

INPUT:
principal interestrate
OUTPUT:
loan amount after 2 years
loan amount after 4 years

EXAMPLE:

INPUT
100 1
OUTPUT:
102
104

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. In each test case, 50% marks are for giving the correct amount after 2 years and 50% marks will be for giving the correct amount after 4 years i.e. if a test case is worth 2 points, 1 point is for correct amount after 2 years and 1 point for correct amount after 4 years.

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.

Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible test cases and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
10 100	30 120
100 5	110 121
100 1	102 104
50 50	100 225
50 200	250 2250
100 0	100 100

Pick your Choice (p3v2d1)

Pick your choice [20 marks]

Problem Statement
We will give you 4 distinct digits from 0-9, i.e. no two digits will be the same. The digits will be given to you as integers in increasing order. For example, we may give you 1, 5, 6, 8. Your job is to find the following

1. Find the largest and second largest single digit number you can form out of the given digits and print their sum. E.g., in the above case, the two largest single digit numbers out of the given digits are 8 and 6, and their sum is 14.

2. Find the largest and second largest double digit number you can form out of the given digits and print their sum. Note that no number should repeat digits but the two numbers may share digits. E.g., in the above case, the two largest double digit numbers out of the given digits are 86 and 85, and their sum is 171. Note that no number repeats digits although the digit 8 is shared across the numbers.
3. Find the largest and second largest triple digit number you can form out of the given digits and print their sum. No number should repeat digits but the two numbers may share digits. E.g., in the above case, the sum of the two largest triple digit numbers out of the given digits is 1726.

Caution

1. Print all three numbers on different lines.
2. The four digits given to you are distinct and are given in increasing order.
3. The two double digit numbers you construct should not repeat digits but the two numbers may share digits.
4. The two triple digit numbers you construct should not repeat digits but the two numbers may share digits.
5. Do not use any datatype other than int.
6. Do not use any library other than stdio.h

HINTS: Visible test cases are there to show you how to give the output as well as warn you if you have extra spaces or extra lines etc in your output.

INPUT:
digit1 digit2 digit3 digit4
OUTPUT:
sum1
sum2
sum3

EXAMPLE:
INPUT
1 5 6 8
OUTPUT:
14
171
1726

Grading Scheme:
Total marks: **[20 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 marks. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible test cases and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
1 5 6 8	14 171

	1726
0 1 2 3	5 63 641
1 2 3 4	7 85 863
3 7 8 9	17 195 1970
2 3 8 9	17 191 1965
6 7 8 9	17 195 1973

Lego Safe (p3v3d1)

Lego Safe [10 marks]

Problem Statement
Your friend wants to give you a secret digit. The digit is confidential and he would like you to store it in a safe. This safe has to be made using lego blocks. To assure him that youâ€™ll keep the digit perfectly secret, you have to send him a image of the safe with the number kept inside it. Following is an empty secure safe.

```
/---\
|%%%|
|% %|
|%%%|
\---/
```

If the number 5 is kept inside, the safe should look like the following

```
/---\
|%%%|
|%5%|
|%%%|
\---/
```

Caution

1. Take good care not to have extra/missing spaces or extra/missing lines in your output. These will fail an exact match and the autograder will give you a zero.
2. Do not use any library other than stdio.h
3. The code to take in an input (using scanf) has already been provided to you.

HINTS: Use the visible test cases to make sure you do not have any extra/missing lines or extra/missing spaces.

EXAMPLE:
INPUT:
5

OUTPUT:
/---\
|%%%|
|%5%|
|%%%|
\---/

Grading Scheme:
Total marks: **[10 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 marks. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 2 points and each hidden test case is worth 4 points. There is 1 visible and 2 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
5	/---\ |%%%| |%5%| |%%%| \---/
0	/---\ |%%%| |%0%| |%%%| \---/
2	/---\ |%%%| |%2%| |%%%| \---/

Race Car (p4v1d1)

Race Car [20 marks]

Problem Statement
Let us recall the equations of motion. If u is initial velocity, a is acceleration, t is time spent, s is displacement, and v is final velocity, then we have

1. v = u + a * t
2. s = u*t + 1/2*a*t*t
3. v*v = u*u + 2*a*s

Your car is driving at current speed u m/sec. You decide to accelerate at a m/sec for t seconds. Your job is to output your velocity at the end of t seconds as well as your displacement in these t seconds. If he velocity or displacement answers are not integers, output just the integer part of the answer. For example, if your velocity is 78.23 m/sec, you should output 78. Similarly, if your displacement is 114.79 m, you should output 114. Print both outputs on different lines.

Caution

1. Do not use math.h or any header file other than stdio.h
2. Use only integer variables. No floats, doubles etc.
3. Do not use loops or conditional statements.
4. Be careful about missing/extra spaces and missing/extra lines.

HINTS:

1. Visible test case number 1 is there to confirm if you are giving output in the correct format or not. Be careful about extra spaces and lines.
2. Visible test case number 2 is there to check if you are performing integer arithmetic operations properly or not.

INPUT:
u a t
OUTPUT:
velocity after t seconds
displacement during these t seconds

EXAMPLE:
INPUT
1 1 1
OUTPUT:
2
1

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. In each test case, 50% marks are for giving the correct velocity and 50% marks will be for giving the correct displacement i.e. if a test case is worth 2 points, 1 point is for correct velocity and 1 point for correct displacement.

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.

Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible test cases and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
1 1 1	2 1
2 1 5	7 22
10 10 10	110 600
10 3 5	25 87
2 10 3	32 51
2 7 1	9 5

Reverse Gear (p4v2d1)

Reverse Gear [20 marks]

Problem Statement
You will be given two positive integers, greater than or equal to 100, but strictly less than 1000 (i.e. these will be 3 digit numbers). For example, we give you 123 and 456. Your first job is to find the digits of the first number. These are 1, 2 and 3. Next, your next job is to compute the number formed if we write these digits in reverse order i.e. in this case it will be the number 321. Your next job is to compute the sum of this reversed number and the second number. In this case, the answer is 321 + 456 = 777.
Print the digits of the first number (in their original i.e. unreversed order) on 3 different lines followed by the sum of the reversed number and the second number on a different line. For example, in the above case, you have to print
1
2
3
777

Caution

1. Print all four outputs on four different lines.
2. The numbers you are given may contain zero. Be careful while reversing. For example, if the two numbers are 300 and 500, then after reversing the first number we will get the number 003 which is just 3. In this case the output should be
   3
   0
   0
   503
3. Do not use any datatype other than int.
4. Do not use any library other than stdio.h

HINTS: Visible test cases are there to show you how to give the output as well as warn you if you have extra spaces or extra lines etc in your output.

INPUT:
number1 number2
OUTPUT:
digit1
digit2
digit3
sum

EXAMPLE:
INPUT
300 500
OUTPUT:
3
0
0
503

Grading Scheme:
Total marks: **[20 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 marks. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible test cases and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
101 202	1 0 1 303
123 456	1 2 3 777
400 700	4 0 0 704
456 987	4 5 6 1641
203 876	2 0 3 1178

222 333	2 2 2 555

Numerical Flowers (p4v3d1)

Numerical Flowers [10 marks]

Problem Statement
We will give you two single digit non-negative numbers, i.e. between 0-9. Suppose we give you the digits 3 and 0. Then your job is to print the following flowery pattern.

```
0 0 0
\ | /
---3---
/ | \
0 0 0
```

We can reveal that there are 3 spaces between the zeros.

Caution

1. Take good care not to have extra/missing spaces or extra/missing lines in your output. These will fail an exact match and the autograder will give you a zero.
2. Do not use any library other than stdio.h
3. The code to take in an input (using scanf) has already been provided to you.

HINTS: Use the visible test cases to make sure you do not have any extra/missing lines or extra/missing spaces.

EXAMPLE:
INPUT:
3 0

OUTPUT:

```
0 0 0
\ | /
---3---
/ | \
0 0 0
```

Grading Scheme:
Total marks: **[10 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 marks. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 2 points and each hidden test case is worth 4 points. There is 1 visible and 2 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
3 0	0 0 0 \ | / ---3--- / | \ 0 0 0
1 7	7 7 7 \ | / ---1--- / | \ 7 7 7
8 8	8 8 8 \ | / ---8--- / | \ 8 8 8