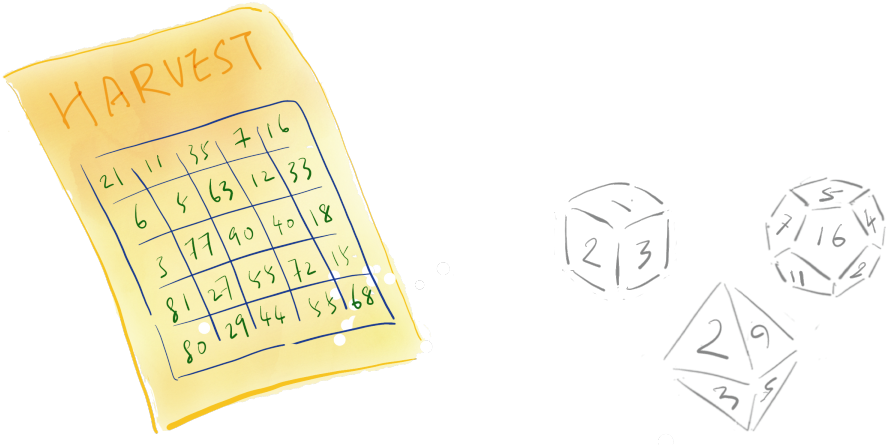
ENGG1330 2N Computer Programming I (19-20 Semester 2)

Assignment 1

Due date: 6-May-2020, 23:59. Late submission: 10% discount per day.



Every year a competition is held in Farmer Peter's village to celebrate the harvest. The prize of this year is a console game package which is one of the best sellers recently and currently out of stock. Peter loves this game package very much and want to win the competition.

The competition is based on a game similar to Bingo. Each participant will get a Bingo card with 25 random numbers arranged in a 5 by 5 grid. The competition consists of indefinite number of rounds. At each round, 3 dice will be rolled and participants can cross out the numbers on their cards if the numbers can be calculated from the numbers rolled with addition, subtraction, division or multiplication. The dice used in the competition can be more than 6 faces.

The calculation must conform to the following format:

number = $dice_1$ op_1 $dice_2$ op_2 $dice_3$

where:

- op_1 and op_2 can be +, -, * or /. E.g. op_1 = + and op_2 = /
- $dice_1$, $dice_2$ and $dice_3$ can be any combination of the numbers rolled at the current round. The value of a die can be used once in the calculation.
- For the division, the divisor must be a factor of the dividend
- Parentheses can be added to override the precedence of operators op1 and op2. i.e. ($dice_1$ op_1 $dice_2$) op_2 $dice_3$ or $dice_1$ op_1 ($dice_2$ op_2 $dice_3$)

E.g. Suppose the numbers rolled at a particular round are 3, 4 and 1. The numbers can be crossed out on the Bingo card should be 1, 2, 6, 7, 8, 9, 11, 12, 13, 15 and 16.

To have better preparation, Peter wants to have a computer program for training. He seeks for your help to write the program with different difficult levels.

A. Level One (10%)

In this level, the program will use the calculation mentioned above to generate all possible numbers from the values of 2 dice, i.e. without op_2 and $dice_3$.

Input: Two integers separated by a space, which are the values of 2 dice.
e.g. 3 9

Output: A list of numbers and the corresponding calculations, sorted by the number and then the calculation. Only number greater than zero should be printed. Each line should be terminated by a line feed. The number and its calculation are separated by a ":". In this part, duplicate expression should be printed (for division only). For example, if 2, 2 are rolled, there should be two 2/2 expressions in the list. The order of operands in the calculation should be same as the input order for +, - and * operators.

Case	Sample Input	Sample Output
1	5 2	3:5-2 7:5+2 10:5*2
2	2 8	4:8/2 6:8-2 10:2+8 16:2*8
3	2 2	1:2/2 1:2/2 4:2*2 4:2+2
4	12 8	4:12-8 20:12+8 96:12*8
5	18 9	2:18/9 9:18-9 27:18+9 162:18*9

B. Level Two (30%)

The program for this level is similar to level one but 3 dice are rolled.

Input: Three integers separated by a space, which are the numbers given by rolling 3 dice.
e.g. 3 9 1
Output: A list of possible numbers separated by a space. The output should be terminated by a space (instead of line feed). Duplicate number should be removed.

Case	Sample Input	Sample Output
1	11 5 2	1 3 4 8 12 14 17 18 21 27 32 33 45 53 57 65 77 110
2	1 1 1	1 2 3

3	18 7 13	2 3 12 24 35 38 73 108 109 113 139 143 217 227 241 325 360 1638
4	9 8 21	4 20 21 22 38 51 93 96 117 159 177 181 197 240 261 357 1512

C. Level Three (30%)

In this level, the program will first read a 5x5 bingo card, a roll (round) count and *n* rows of strings. Each string contains 3 numbers as in level two. The program should calculate all the possible numbers from 3 dice of each roll and cross out the corresponding numbers on the bingo card. The number crossed out should be replaced with a "x" mark.

Input:

6 + n lines of inputs

1st - 5th lines	The numbers on the 5 by 5 bingo card. Each line contains 5 integers separated by a space.
6th line	Integer, n, numbers of rolls to be inputted.
7th - (7+n-1)th lines	n rows of roll. Each row contains 3 integers separated by a space.

Output:

The numbers/marks on the bingo card in 5 by 5 grid. Each line should be terminated by a line-feed character including the last line. Numbers/marks will be printed in 4 characters width with right alignment.

Case	Sample Input	Sample Output
1	1 2 3 4 5 11 10 13 14 15 26 28 29 25 21 5 3 2 1 4 15 13 14 21 26 4 5 1 3 6 11 8 6 5 4 10 1 3	x x x x x 11 x x x x x 28 x x 21 x x x x x x x x 21 x
2	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 1 10 5 2	x 2 x x 5 6 x 8 9 x 11 12 x 14 x 16 x 18 19 x 21 22 23 24 x
3	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 1 1 1 1	x x x 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
4	1 2 3 4 5 5 2 1 3 4 4 3 1 2 5 3 5 2 1 4 1 2 2 3 4 2 1 1 1 1 1 1	x x x 4 5 5 x x x 4 4 x x x 5 x 5 x x 4 x x x x 4

D. Level Four (30%)

In this level, your program will find out at which roll (start from one) Bingo can be claim. The program should read all the given input before the program end.

Input: Same as Level 3
Output:
Same as Level 3 if no Bingo could be claimed,
Otherwise, print out a string "Bingo! At round:*N*", where *N* is the N^{th} roll (start from 1 to N) which lead to Bingo, followed by the snapshot of the bingo card at time of Bingo.
In any case, only one bingo card will be printed.

Case	Sample Input	Sample Output
1	1 2 3 4 5 11 10 13 14 15 26 28 29 25 21 5 3 2 1 4 15 13 14 21 26 4 5 1 3 6 11 8 6 5 4 10 1 3	Bingo! At round:3 x x x x x 11 x x x x x 28 x x 21 x x x x x x x x 21 x
2	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 1 10 5 2	x 2 x x 5 6 x 8 9 x 11 12 x 14 x 16 x 18 19 x 21 22 23 24 x
3	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 2 10 5 2 5 1 1	Bingo! At round:2 x 2 x x x x x 8 9 x 11 12 x 14 x 16 x 18 19 x 21 22 23 24 x
4	1 2 3 4 5 5 2 1 3 4 4 3 1 2 5 3 5 2 1 4 1 2 2 3 4 2 1 1 1 1 1 1	Bingo! At round:1 x x x 4 5 5 x x x 4 4 x x x 5 x 5 x x 4 x x x x 4
5	1 2 3 4 5 5 2 1 3 4 4 3 1 2 5 3 5 2 1 4 1 2 2 3 4 3 1 1 1 1 1 1 10 2 4	Bingo! At round:1 x x x 4 5 5 x x x 4 4 x x x 5 x 5 x x 4 x x x x 4

II. Submission

Virtual Programming Lab (VPL) will be setup for testing and submission. Each part will have two VPLs, one for testing and one for final submission. You can only submit your program once to the final submission VPL but submission to the testing VPL is unlimited. Please well test your program on the testing VPL before submission. Your program should generate the output based on the specifications, i.e. without extra text/space. We may mark your program with test cases different from those found in VPL.

Programming style and efficiency will be part of the assessment.

Program should only use the Python libraries/features covered in this course. Using of 3rd party library is not allowed.

Plagiarism (detected by the system) will get zero marks, apply to the source providers as well.