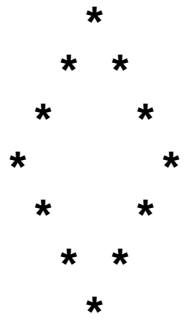
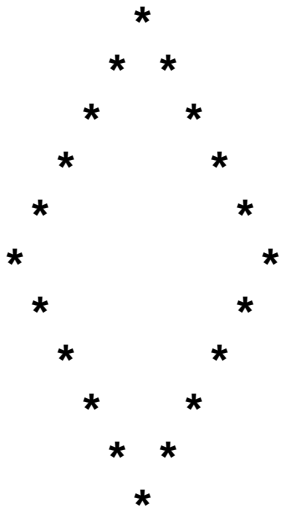
D1 Batch - Mock Quiz (13th Feb 2025)

Q0) Hollow Diamonds

This question is pretty straightforward, given an **integer n,** print out a **hollow diamond** pattern, spread over n lines
Make sure to calculate the spaces and asterisks (*) for each line correctly! Make sure that the hollow structure is maintained.
Execute the code on your computer's terminal to see the output and then submit on Bodhitree. Incase you are not sure of how to use the terminal, please ask the TAs.

E.g.

n = 7	n = 11
	

Input Format

- A single integer n

Output Format

- A symmetric, hollow diamond pattern with n rows. The first and last row contain a single *.

Assumptions on Input

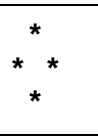
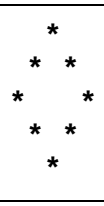
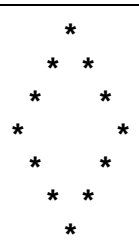
- Assume that the value of n entered by the user is an **odd number** ranging **from 3 to 25**.

Note:

- Do not write any C++ statements for printing general messages. For example, the following should NOT be present in your program:
  - cout << "Enter a number:",
  - cout << "The computed answer is", etc.

- cout should be used to print only the computed final output. In addition, do not print unnecessary spaces unless specified in the program.
- If any hard coding is found, or if any test case passes by merely writing a cout statement and without any logic, then the marks for that test case will NOT be awarded.

Visible Test Cases :

No.	Input	Output
1	3	
2	5	
3	7	

Q1) Shelf Sorting

Your bookshelf is a mess. Since you're not an avid reader (apologies if you are), you only have **4 books** on your bookshelf, and the book names are only **one character long** and only consist of **uppercase letters**. You decide to place the books **sorted in alphabetical order**.

Write an algorithm that will **sort** out the **4 characters** and **print** them in **ascending order**

Write a function **sortBooks**, that takes in **4** characters **passed by reference,** which **returns void**, and sorts the characters in ascending order

To solve this question, we will use a trivial version of the bubble sort algorithm:

- Assume that the 4 characters are stored in variables called a, b, c, d
- We'll be **comparing** the **ASCII values** of the variables for sorting them

First Pass:

1. If a is bigger than b, swap them
2. If b is bigger than c, swap them
3. If c is bigger than d, swap them

Second Pass:

1. If a is bigger than b, swap them
2. If b is bigger than c, swap them

Third Pass:

1. If a is bigger than b, swap them

Note that after the first pass, **d will contain the biggest character out of a, b, c, d.**
Now, in the second pass, you will put the second biggest character in c, and the third pass will put the third biggest character in b, and a will get the smallest character for free!

Example: Let a = 'H', b = 'Z', c = 'D', d = 'K'
First Pass:

1. a and b remain as it is
2. b and c get swapped, and we have b = 'D', c = 'Z'
3. c and d get swapped, and we have c = 'K', d = 'Z'

Now, we have a = 'H', b = 'D', c = 'K', d = 'Z'
Second Pass:

1. a and b get swapped, we have a = 'D', b = 'H'
2. b and c remain as it is

We have a = 'D', b = 'H', c = 'K', d = 'Z'
Third Pass:

1. a and b remain as it is

At the end, you will print out: **DHKZ**
We have successfully sorted the four books!

Convince yourself that after 3 passes, these 4 characters will be sorted. Now think if you had **n characters**, you would **require n-1 such passes** to **sort** all these characters, ask a TA if this doesn't make sense to you.

Input Format:

- The first line contains an integer t, the number of test cases
- The next t lines each contain 4 book names, a, b, c, d the four characters to be sorted

Output Format:

- Output the four characters, a, b, c and d, which should be printed out in the sorted order

Assumptions on input:

- 0 <= t <= 1e9
- a, b, c, d will be **uppercase letters**

Note:

- Do not write any C++ statements for printing general messages. For example, the following should NOT be present in your program:
  - cout << "Enter a number:",
  - cout << "The computed answer is", etc.
- cout should be used to print only the computed final output. In addition, do not print unnecessary spaces unless specified in the program.
- If any hard coding is found, or if any test case passes by merely writing a cout statement and without any logic, then the marks for that test case will NOT be awarded.

Visible Test Cases :

No.	Input	Output
1	2 J K I L V T U S	IJKL STUV
2	3 M N O P X Y A B G E F H	MNOP ABXY EFGH
3	4 D C B A H G I F Y X W Z M O N P	ABCD FGHI WXYZ MNOP

Q2) Problem Statement: Fanum Tax

When Om was deep asleep, he saw the union budget meeting going on in his dreams.
The tax policy was defined as follows:

- If a person with x years of experience earns a salary of y, the xth bit of y determines whether they are taxed.
- If the xth bit of y is 0, the person is **not taxed**.
- If the xth bit of y is 1, the person is **taxed** an amount equal to 2^(x-1), and their new salary will be updated as y - 2^(x-1). This is equivalent to setting the xth bit of y to 0.

Om's friends and family started reaching out to him for tax calculations. Help him determine whether a person will be taxed and compute their net income accordingly.

Write a function **calculateNetIncome,** which takes in two arguments.

- An integer x, the years of experience
- An integer y, the salary of the person, **as a reference**

The function will **return a boolean** value, whether the person will be taxed or not.

- If the xth bit of y is zero, the function will return false, else the xth bit will be set to zero, and the function will return true
- **Use bit shift and bitwise operators** (>> , &) for this, instead of arithmetic operators (*, +)
- Try to **avoid using loops**

The main program will print "Not Taxed" on the first line, if the person wasn't taxed, else will print "Taxed" and the second line will print the value of y.

Input Format

- A single integer t, the number of test cases.
- For each test case, two integers:
    - x: The years of experience.
    - y: The salary of the person.

Output Format:

- The first line prints "Taxed" if the person was taxed, else prints "Not Taxed"
- The second line prints the net income of the person, y

Assumptions on input:

- 0 <= t, y <= 1e9
- 1 <= x <= 31

Note:

- Do not write any C++ statements for printing general messages. For example, the following should NOT be present in your program:
    - cout << "Enter a number:",
    - cout << "The computed answer is", etc.

- cout should be used to print only the computed final output. In addition, do not print unnecessary spaces unless specified in the program.
- If any hard coding is found, or if any test case passes by merely writing a cout statement and without any logic, then the marks for that test case will NOT be awarded.

Visible Test Cases :

No.	Input	Output
1	2 5 31 3 15	Taxed 15 Taxed 11
2	3 2 10 3 8 1 7	Taxed 8 Not Taxed 8 Taxed 6
3	1 31 2147483647	Taxed 1073741823

Q3) Time to Crib!

You are attending the **CS101 crib session**, and you need to **quickly check** whether your **marks add up to the total** written on the first page. Since you are **very lazy**, you decide to write a program to **sum up the marks**, and to check whether the sum is right.

Write a function **sumMarks** that will take an **integer n** as an argument, denoting the number of questions, and returns a **double,** containing the sum of marks of n questions.

Try to use **recursion** to solve this problem.

The base case would be when **n = 0**, you will **return 0.0**, and for the recursive step, you will take as **input a double m**, and **call the function recursively** to find the sum of the remaining n-1 numbers, finally **adding in m** to this, and returning this value.

The main program would call sumMarks, and then check if the sum is equal to the total marks written at the front. If they are equal, print “Total is Correct!”, otherwise print “Time to Crib!”

Input Format:

- An integer n and a double totalMarks, denoting the number of questions and total marks awarded
- n double numbers, each denoting the marks awarded for the particular question

Output Format:

- Print “Total is Correct!”, if the sum is equal to totalMarks, else print “Time to Crib!” if they aren’t equal

Assumptions on input:

- 0 <= n <= 1e5
- 0 <= totalMarks <= 1e9
- 0 <= mark <= 1e5
- Note that the sum of marks wont overflow the double data type.

Note:

- Do not write any C++ statements for printing general messages. For example, the following should NOT be present in your program:
    - cout << "Enter a number:",
    - cout << "The computed answer is", etc.
- cout should be used to print only the computed final output. In addition, do not print unnecessary spaces unless specified in the program.
- If any hard coding is found, or if any test case passes by merely writing a cout statement and without any logic, then the marks for that test case will NOT be awarded.

Visible Test Cases :

No.	Input	Output
1	4 6 1.2 1.8 1.5 1.5	Total is Correct!
2	6 4.5 1 2 3.2 1.5 1.3 1.6	Time to Crib!
3	3 4 1.1 2.5 0.4	Total is Correct!

Q4) Calculator

Let's say on a calculator we enter a number, an operator, another number, and press =. We get some answers. Then, we just press the mathematical operator and a number, which operates on the answer obtained from the previous result.

E.g.
21 + 9 = 30
- 5 = 25 (i.e. 30 - 5)
* 2 = 50 (i.e. 25 * 2)

We need to simulate this behavior.

Write a C++ program as follows:

Function:
- Write a function 'calculator' that accepts three arguments: two integers and one character. Based on the character, compute addition, subtraction, multiplication, and division. Return the result in the main function

Main Program:
1. Accept an integer from the user, followed by a mathematical operator + - * / and followed by another integer
2. Pass the numbers and operator to the function
3. Get the result and store it
4. Accept another mathematical operator and an integer
5. Pass the result obtained in point 3 along with the operator and number obtained in point 4 to the function
6. Get the result and store it.
7. Do this in a loop i.e. Point 4, 5, 6, till the user enters X as the mathematical operator
8. In the end, print the result

Example
Input
100
+ 5
- 8
* 3
+ 7
/ 2
X

Output
149

Input Format
- The first line contains an integer
- The subsequent lines contain a mathematical operator followed by an integer, separated by a single space

Output Format
- An integer

Assumptions on Input

- Assume that the integers input by the user will be between -1000 and 1000
- Assume that valid mathematical operators will be added +, -, *, /
- Assume that no invalid operations can be performed (i.e. division by zero)

Visible Test Cases :

No.	Input	Output
1	-999 + 1000 / 2 * -5 - 10 * 3 X	-30
2	500 * 2 * -1 / 3 / 5 / 2 * -300 / -5 X	-1980
3	-2 * -3 - -5 * -10 / -8 * 5 + -90 X	-25

Q5) Rectangles: Find overlapping area

You are given the coordinates of two rectangles, and each rectangle is defined by the coordinates of its top-left and bottom-right corners. The task is to calculate the area of the overlapping region between the two rectangles. If the rectangles do not overlap, return 0.

Write a C++ program that accepts 8 integers as input from the user. These represent the coordinates of the two rectangles: (x1, y1), (x2, y2), (x3, y3), and (x4, y4). Pass these to a function getArea that returns the **intersected area** of both rectangles

Input Format:
You are given two rectangles. Each rectangle is represented by four integers:

- The first line contains four integers which represent the top-left and bottom-right coordinates of the first rectangle
- The second line contains four integers which represent the top-left and bottom-right coordinates of the first rectangle

Output Format:

- Return the area of the overlapping region. If there is no overlap, return 0.

Assumptions on input:

- Assume that the value entered by the user for the coordinates is between 1 and 100, both inclusive
- Assume that the coordinates will always form a valid rectangle

Note:

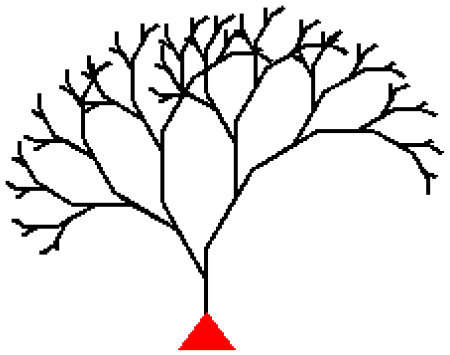
- Do not write any C++ statements for printing general messages. For example, the following should NOT be present in your program:
    - cout << "Enter a number:",
    - cout << "The computed answer is", etc.
- cout should be used to print only the computed final output. In addition, do not print unnecessary spaces unless specified in the program.
- If any hard coding is found, or if any test case passes by merely writing a cout statement and without any logic, then the marks for that test case will NOT be awarded.

Visible Test Cases :

No.	Input	Output
1	1 5 5 1 3 6 7 2	6
2	1 5 7 1 2 4 6 2	8
3	1 5 3 3 4 6 6 4	0

Q6) Branching Out (Optional) (Tree using recursion)

Your job is to make a tree similar to the one given below in simplecpp. Write a function **tree that** takes in the depth of the tree, n, and makes this tree with a depth of n. Think in terms of recursion, on how this tree can be made.



Input Format:

- A single integer n

Output Format:

- A tree

Assumptions on input:

- 1 <= n <= 10

Note:

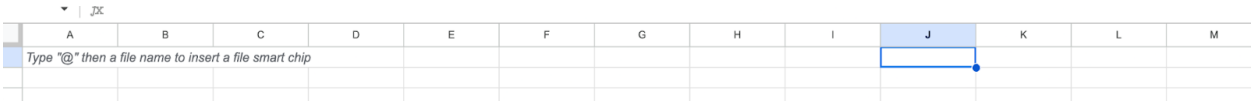
- Do not write any C++ statements for printing general messages. For example, the following should NOT be present in your program:
  - cout << "Enter a number:",
  - cout << "The computed answer is", etc.
- cout should be used to print only the computed final output. In addition, do not print unnecessary spaces unless specified in the program.
- If any hard coding is found, or if any test case passes by merely writing a cout statement and without any logic, then the marks for that test case will NOT be awarded.

Visible Test Cases :

No test cases for this question

Q7) Sheet Happens (Optional)

Om is bored of using google sheets, it doesn't match his freak. He plans on creating his own sheets in c++. Help him in giving column names, as they are in sheets.



Given a number n, convert it to the corresponding column name. Write a function excellentColumn that takes in an **integer n,** and returns **void,** and prints out the column name (recursively?)

For example:
N = 1 => A
N = 10 => J
N = 29 => AC
N = 52 => AZ
N = 5103 => GNG

Think about this in terms of recursion. Read the algorithm given below if you are having trouble figuring it out yourself.

Base Case:
If n is equal to 0, we will simply return, without printing anything
Otherwise:
Call the function excellentColumn **recursively**, with the **argument being (n-1)/26**
Print out the correct number corresponding to **(n-1)%26,** (0 would map to A, 1 would map to B, and so on)

Figure out why this code works, otherwise **ask a TA**

Input Format:

- A number t, the number of test cases
- A single integer n, for each of the t test cases

Output Format:

- A name for each of the column, separated by new lines (t lines with their corresponding column names)

Assumptions on input:

- 1 <= n <= 1e9
- 1 <= t <= 10

Note:

- Do not write any C++ statements for printing general messages. For example, the following should NOT be present in your program:
    - cout << "Enter a number:",
    - cout << "The computed answer is", etc.
- cout should be used to print only the computed final output. In addition, do not print unnecessary spaces unless specified in the program.
- If any hard coding is found, or if any test case passes by merely writing a cout statement and without any logic, then the marks for that test case will NOT be awarded.

Visible Test Cases :

No.	**Input**	**Output**
1	30	AD
2	502	SH
3	1165	ARU