D2 Batch - Mock Quiz (11st 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 *.

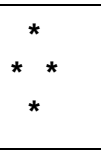
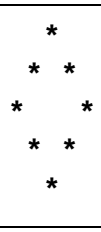
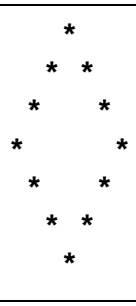
Assumptions on Input

- 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. UNO, DOS, SORT

You are playing UNO with your friends, and are down to your last **four cards.** Whoever wins the game will get an ice-cream sponsored by the other players!

In order to get a winning edge on your friends, you want to **sort** your cards in **ascending order**. We assume there are no colors, and we extend the card numbers to all integers.

Given 4 integers, your job is to **sort** the 4 numbers, and **print** them in the sorted order. Write a function **sortCards**, that takes in **4** integers **passed by reference,** which **returns void**, and sorts the numbers in ascending order. If required, the sortCards function uses a **swap** function for swapping two integers. The inputs to swap function are **passed by reference**, which **returns void**, and swaps the two input integers. (Refer to lecture-9 slides for details on swap function).

The main function accepts an integer **'t'**, the number of test cases. For each test case, it takes in 4 integers, **a, b, c, d**, and calls the sortCards function. All 4 integers are passed to the sortCards function by reference. Remember, **the final output for each test case should be printed from the main function only.**

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

- Assume that the 4 numbers are stored in variables called a, b, c, d

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 number out of a, b, c, d.**
Now, in the second pass, you will put the second biggest number in c, and the third pass will put the third biggest number in b, and a will get the smallest number for free!

Example: Let a = 4, b = 20, c = 1, d = 5

First Pass:

1. a and b remain as it is
2. b and c get swapped, and we have b = 1, c = 20
3. c and d get swapped, and we have c = 5, d = 20

Now, we have a = 4, b = 1, c = 5, d = 20

Second Pass:

1. a and b get swapped, we have a = 1, b = 4
2. b and c remain as it is

We have a = 1, b = 4, c = 5, d = 20

Third Pass:

1. a and b remain as it is

We have successfully sorted the four numbers!

Convince yourself that after 3 passes, these 4 numbers will be sorted. Now think if you had **n numbers**, you would **require n-1 such passes** to **sort** all these numbers, 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 integers, **a, b, c, d,** the four numbers to be sorted

Output Format:

- **'t'** lines are printed
- Each line outputs the four numbers, a, b, c and d, which should be printed out in the sorted order

Assumptions on input:

- 0 <= t, a, b, c, d <= 1e9

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 12 7 9 1 56 23 0 8 1299 12 70 1388	1 7 9 12 0 8 23 56 12 70 1299 1388
2	1 90 87 534 45567	87 90 534 45567
3	4 198 209 313 460 87 76 43 34 5 6 5 6 3 8 4 9	198 209 313 460 34 43 76 87 5 5 6 6 3 4 8 9

Q2) Count number of 1's

Given an integer n, your task is to determine the number of 1's in its binary representation. For example, the binary representation of 178 is 10110010. The output is 4 as there are four 1's.

Write a C++ program to accept the integer 'n', in the main, pass it to a function called **popcount,** and **returns the number of ones** in the binary representation of n. Print the result in the main function.

Input Format:

- A single integer n (0 <= n <= 10^9)

Output Format:

- A single integer representing the count of 1's in the binary representation of n.

Assumptions on input:

- N is a non-negative integer in the range (0 <= n <= 10^9).
- Input is a single, well-formatted decimal integer.
- No leading zeros, non-numeric values, or special characters.

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**	**Binary Equivalent**
1	5	2	101
2	170	4	10101010
3	1000000000	13	1110111001101011001010000000000

Q3) Reverse a number using Recursion

Write a C++ program to accept an integer N from the user in the main function. Write a recursive function that reverses this integer N. The reversed number should be printed in the main function.
For ex : N=345 output is 543.

Input Format:

- A single integer N (0 <= N <= 10^9)

Output Format:

- A single integer representing the reversed number.

Assumptions on input:

- N is a valid integer in the range (0 <= N <= 10^9).
- The integer will not begin or end with a zero

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	2154	4512
2	101	101
3	6	6

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:

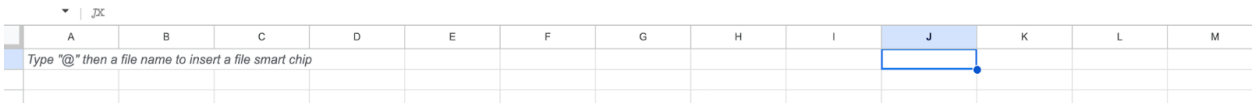
- Do not write any C++ statements for printing general messages. For example, the following should NOT be present in your program:
    - cout << "Enter a number:",
    - cout << "The computed answer is", etc.
- cout should be used to print only the computed final output. In addition, do not print unnecessary 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) Sheet Happens (Optional) (Given an integer generate excel header 27 > AA)

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:

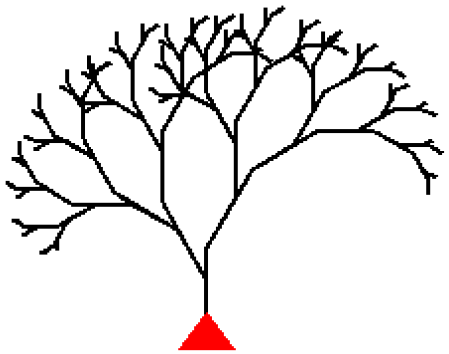
- Do not write any C++ statements for printing general messages. For example, the following should NOT be present in your program:
  - cout << "Enter a number:",
  - cout << "The computed answer is", etc.
- cout should be used to print only the computed final output. In addition, do not print unnecessary 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

Q7. 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