Question **1**

Correct

Mark 1.00 out of 1.00

In this challenge, you will learn to implement the basic functionalities of pointers in C. A pointer in C is a way to share a memory address among different contexts (primarily functions). They are primarily used whenever a function needs to modify the content of a variable, of which it doesn't have ownership.

In order to access the memory address of a variable, , we need to prepend it with $\&$ sign. E.g., `&val` returns the memory address of val.

This memory address is assigned to a pointer and can be shared among various functions. E.g. $int^{*}p = \&val$ will assign the memory address of val to pointer p. To access the content of the memory to which the pointer points, prepend it with a `*`. For example, `*p` will return the value reflected by val and any modification to it will be reflected at the source (val)

```
    void increment(int *v) {
        (*v)++;
    }

    int main() {
        int a;
        scanf("%d", &a);
        increment(&a);
        printf("%d", a);
        return 0;
    }
```

You have to complete the function `void update(int *a,int *b)`, which reads two integers as argument, and sets a with the sum of them, and b with the absolute difference of them if the result of a+b is even or the product of them if a+b is odd.

- *a'=a+b*
- if a' is even: *b'=|a-b|*
- if a' is odd: *b'=a*b*

Input Format

The input will contain two integers, a and b, separated by a newline.

Output Format

You have to print the updated value of a and b, on two different lines.

Sample Input

```
4
5
```

Sample Output

```
9
20
```

Explanation

- a'=4+5=9 //a' is odd
- b'=4*5=20

Answer: (penalty regime: 0 %)

Reset answer

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
#include <stdio.h>
#include <stdlib.h>

void update(int *a,int *b) {
    int tmp_a = *a;
    *a = *a + *b;
    if(*a%2 == 0) {
        *b = abs(tmp_a - *b);
    }
    else {
        *b = tmp_a * *b;
    }
}

int main() {
    int a, b;
    int *pa = &a, *pb = &b;
```

	Input	Expected	Got	
✓	4 5	9 20	9 20	✓
✓	4 6	10 2	10 2	✓
✓	1 1	2 0	2 0	✓
✓	230 150	380 80	380 80	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

In mathematics, the Fibonacci numbers are the numbers in the following integer sequence, called the Fibonacci sequence, and characterized by the fact that every number after the first two is the sum of the two preceding ones:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377...

By definition, the first two numbers in the Fibonacci sequence are either 1 and 1, or 0 and 1, depending on the chosen starting point of the sequence, and each subsequent number is the sum of the previous two.

This series can be broken down as the following series:

- Fib0 = 0
- Fib1 = 1
- Fibn = Fibn-1 + Fibn-2 , n > 1

Tribonacci sequence:

The tribonacci sequence is a generalization of the Fibonacci sequence where each term is the sum of the **three preceding** terms. This series can be broken down as the following series:

- Trib0= 0
- Trib1= 1
- Trib2= 1
- Tribn= Tribn-1 + Tribn-2 + Tribn-3, n>2

The sequence begins

You will create an array smaller than n using malloc/calloc to store the values of the series on the heap. You will use realloc to resize the array as needed. **You will recompute the Tribonacci series with every iteration (even if you already have the values).**

Input Format

Input will contain one integer, n, as the amount of series to compute and print.

Output Format

You will print ***n*** tribonacci series up to ***i*** where ***{i|0<=i<=n}*** in new lines separeted by a single space. Each new line will print the tribonacci series for ***i+1***.

Sample Input 0

```
6
```

Sample Output 0

```
0
0 1
0 1 1
0 1 1 2
0 1 1 2 4
0 1 1 2 4 7
```

Answer: (penalty regime: 0 %)

Reset answer

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
#include <stdio.h>
#include <stdlib.h>

// Calculates the n-th element of the tribonacci sequence. The start index being '1' instead of '0'
int tribonacci_calculation(int n);

//n=amount of numbers in the series to compute, seq=array to store series
void tribonacci(int n, int* seq){
    switch (n) {
        //Store base cases in seq or calculate the n-th element in sequence and store.
        case 1:
            *seq = 0;
            break;
        case 2:
            *seq = 0;
            *(seq+1) = 1;
            break;
        case 3:
```

	Input	Expected	Got	
✓	5	0 0 1 0 1 1 0 1 1 2 0 1 1 2 4	0 0 1 0 1 1 0 1 1 2 0 1 1 2 4	✓
✓	2	0 0 1	0 0 1	✓
✓	6	0 0 1 0 1 1 0 1 1 2 0 1 1 2 4 0 1 1 2 4 7	0 0 1 0 1 1 0 1 1 2 0 1 1 2 4 0 1 1 2 4 7	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

In this problem, you will implement a decoder that produces strings from ciphertexts by:

1. Converting the string into an nxn matrix in column-major order
2. Generating the new string by reading the matrix in row major order

For example, the following matrix would be constructed for the input string "WECGEWHYAAIORTNU":

- W E A R
- E W A T
- C H I N
- G Y O U

Taking characters in row-major order from this matrix, the decrypted text would read:

- WEAREWATCHINGYOU

Input Format

Input starts with a line consisting of a single number T. T test cases follow. Each test case consists of one line. This line contains the cipher text. The cipher text contains either UPPERCASE letters or blank spaces.

Constraints

Total number of character in the text will not be more 10,000.

Output Format

For each test case, the output contains a single line containing the decrypted text. If the number of characters in the input text is not square of any number, then give the output 'INVALID'

Sample Input 0

```
3
WECGEWHYAAIORTNU
DAVINCICODE
DTFRIAOEGLRSI TS
```

Sample Output 0

```
WEAREWATCHINGYOU
INVALID
DIGITAL FORTRESS
```

Sample Input 1

```
1
DIIANCVCO
```

Sample Output 1

```
DAVINCICO
```

Sample Input 2

```
1
IOCM
```

Sample Output 2

```
ICOM
```

Answer: (penalty regime: 0 %)

Reset answer

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
// Library imports
#include <stdio.h>
#include <string.h>
#include <math.h>
#include<stdbool.h>

// Variable definitions
#define MAX_CHARS 10000

/*
 * CypherText Decoder
 *   In this problem, you will implement a decoder that produces strings from ciphertexts by:
 *      1. Converting the string into an nxn matrix in column-major order
 *      2. Generating the new string by reading the matrix in row major order
 *   For example, the following matrix would be constructed for the input string "WECGEWHYAAIORTNU":
 *                                      n | 0 1 2 3
 *                                    n    --------
 *                                    0 |  W E A R
```

	Input	Expected	Got	
✓	4 WE OUE OUT LNGSAO H RWDN S EFEM FNOCEDROE CSERULES TTA ARO H LNEN ISLE VE SUISTI CTSHO TKT EU ESLT S S EHP	WE ARE LOW ON DOUGHNUTS SEND MORE COFFEE INVALID THIS TEST SUCKS A LITTLE LESS THAN THE PREVIOUS ONE	WE ARE LOW ON DOUGHNUTS SEND MORE COFFEE INVALID THIS TEST SUCKS A LITTLE LESS THAN THE PREVIOUS ONE	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

Complete the following script to sort a list of words/numbers.

Use the pointer "word" and the double pointer "word_list" to store the words given by the STD Input.

NOTE: Complete the code following the TODO hints

Answer: (penalty regime: 0 %)

Reset answer

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAX_CHARS 100
#define MAX_LINES 100           /* max #lines to be sorted */



void qsort_ (void *v[], int left, int right, int (*comp) (void *, void *));


int main (int argc, char *argv[])
{

    int nlines;                 /*number of input lines read */

    char *word;
```

	Test	Input	Expected	Got	
✓	1	4 CCC BBB DDD AAA	AAA BBB CCC DDD	AAA BBB CCC DDD	✓
✓	2	5 eeeee aaaaa bbbbb ddddd ccccc	aaaaa bbbbb ccccc ddddd eeeee	aaaaa bbbbb ccccc ddddd eeeee	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.