# CPro-Assignment 3

**Deadline:** 18th September, 2024. 11:59 PM

## 1. Dr. Doofenshmirtz

Phineas and Ferb have designed a high-tech calculator. However, their sister Candance wants to show them wrong and has come up with two numbers, **num1** and **num2**, that they need to multiply. To make things more interesting, they have represented these numbers as strings rather than regular integers. The challenge is to find the product of these two numbers and display it as a string, just like how they like to do things in their imaginative world.

Formally, given two non-negative integers **num1** and **num2** represented as strings, return their product as a string.

**Note:** You must not convert the inputs to integers directly.

### 1.1 Input Format

- Lengths of the two non-negative integers *L1* and *L2* and *num1*, *num2*, each represented as strings.

### 1.2 Output Format

- A string representing the product of *num1* and *num2*.

### 1.3 Examples

#### Example 1:

```
Input:
    1 1
    2 3
Output:
    6
```

#### Example 2:

```
Input:
    3 3
    123 456
Output:
    56088
```

### 1.4 Constraints

- 0 ≤ num1.length, num2.length ≤ 500
- num1 and num2 consist of digits only.
- Both num1 and num2 do not contain any leading zero, except the number 0 itself.

## 2. Breaking Enigma

Professor Turing, a brilliant computer scientist, has published many influential research papers. To measure his academic impact, Professor Turing wants to calculate his **h-index** and **i-index**, a metric that reflects both the number of papers he has published and how often they are cited.

**h-index:** The largest number h such that the professor has published at least h papers, each cited h times or more.

**i-index:** The number of publications with at least 10 citations.

You are required to calculate both the h-index and i-index, using pointers and dynamic memory allocation in C.

### 2.1 Examples

**Example 1:**

```
Input:
    5
    0 1 3 5 6
Output:
    3 0
```

**Explanation:** The researcher has 5 papers in total, receiving 0, 1, 3, 5, and 6 citations, respectively. Since the researcher has 3 papers with at least 3 citations each, their h-index is 3. No paper has 10 or more citations, so the i-index is 0.

**Example 2:**

```
Input:
    3
    1 2 100
Output:
    2 1
```

### 2.2 Input Format

- An array of integers *citations* where each element represents the number of citations for the corresponding paper.

### 2.3 Output Format

- An integer representing the researcher’s h-index and i-index.

### 2.4 Constraints

- n == citations.length
- $1 \le n \le 10^5$
- $0 \le citations[i] \le 1000$
- citations is sorted in ascending order.

---

## 3. Jagged Arrays

Jagged arrays, also known as "ragged arrays," are a type of multidimensional array where the inner arrays can have different lengths. Your task is to implement a 2D jagged array in C by dynamically allocating memory.

### Details:

- You will be given the number of rows **(r)** as input.
- Initially, assume the jagged array is an *r x 1* matrix filled with value 1.
- The jagged array should support two operations:
    - Operation 1: Append a given value *val* to the end of the *i*-th row.
    - Operation 2: Print the entire row *i*.

### 3.1 Input Format

- The first line contains an integer **r** (Number of rows) and **t** (Number of queries).
- For each query, there are two cases possible:
    - Operation 1: You will be given a row **i** and value **val**.

  - Operation 2: You will be given a row **i** to print.

### 3.2 Output Format

- Print the entire row when required.

### 3.3 Sample Test Case

```
Input:
    4 8
    2 2
    1 2 4
    2 2
    1 2 5
    2 2
    2 1
    1 1 3
    2 1

Output:
    1
    1 4
    1 4 5
    1
    1 3
```

## Operations Explanation:

**Initial Matrix:** 4x1 matrix with each row having value 1

```
    1
    1
    1
    1
```

**State of the Array at Each Query:**

## First Query: 2 (operation) 2 (row)

The second row should be printed. The matrix will remain unchanged.

## Second Query: 1 (operation) 2 (row) 4 (val)

The second row would be updated and 4 should be appended to the end of 2nd row. The new matrix is:

```
1
1 4
1
1
```

### Third Query: 2 2

The second row should be printed. The matrix will remain unchanged.

### Fourth Query: 1 2 5

The second row would be updated and 5 should be appended to the end of 2nd row. The new matrix is:

```
1
1 4 5
1
1
```

### Fifth Query: 2 2

The second row should be printed. The matrix will remain unchanged.

### Sixth Query: 2 1

The first row should be printed. The matrix will remain unchanged.

### Seventh Query: 1 1 3

The first row would be updated and 3 should be appended to the end of first row. The new matrix is:

```
1 3
1 4 5
1
1
```

### Eighth Query: 2 1

The first row should be printed. The matrix will remain unchanged.

## 3.4 Notes

- Using static memory for arrays in this problem in any form will result in a straight 0 for the entire assignment. You have to use malloc, calloc, and realloc.
- Number of rows is **r** where 1 ≤ r ≤ 100.
- Number of queries is **q** where 1 ≤ q ≤ 100.

Have fun coding! 😄

help / here.

Okay, 100+ Computer Science Concepts Explained, click here.

## Submission guidelines:

<---- Do not rename any files given in the handout. Only write the code in the specified C (p#/main.c) files in the respective directories ---->