








































# Practice Arena

Practice problems aimed to improve your coding skills.

-  PRACTICE-02\_SCAN-PRINT
-  PRACTICE-03\_TYPES
-  LAB-PRAC-02\_SCAN-PRINT
-  LAB-PRAC-01
-  PRACTICE-04\_COND
-  BONUS-PRAC-02
-  LAB-PRAC-03\_TYPES
-  PRACTICE-05\_COND-LOOPS
-  LAB-PRAC-04\_COND
-  LAB-PRAC-05\_CONDLLOOPS
-  PRACTICE-07\_LOOPS-ARR
-  LAB-PRAC-06\_LOOPS
-  LAB-PRAC-07\_LOOPS-ARR
-  LABEXAM-PRAC-01\_MIDSEM
-  PRACTICE-09\_PTR-MAT
-  LAB-PRAC-08\_ARR-STR
-  PRACTICE-10\_MAT-FUN
-  LAB-PRAC-09\_PTR-MAT
-  LAB-PRAC-10\_MAT-FUN
-  PRACTICE-11\_FUN-PTR
-  LAB-PRAC-11\_FUN-PTR
-  LAB-PRAC-12\_FUN-STRUC
  -  Point Pairing Party
  -  Verify the family tree of Mr C
  -  Simple Sudoku
  -  The Family Tree of Mr C Part Three
  -  The Post offices of KRville
  -  Matrix Mandala
  -  Mango Mania
  -  Recover the Rectangle
  -  Crazy for Candy
  -  A Brutal Cipher Called Brutus
  -  Triangle Tangle
  -  Basic Balanced Bracketing
-  LABEXAM-PRAC-02\_ENDSEM
-  LAB-PRAC-13\_STRUC-NUM
-  LAB-PRAC-14\_SORT-MISC

# Verify the family tree of Mr C

## LAB-PRAC-12\_FUN-STRUC

**Verify the family tree of Mr C [20 marks]**

---

**Problem Statement**

Mr C wants to verify if his family tree has any errors or not. In the first line of the input, you will be given a strictly positive integer  $n$ , denoting the number of family members. In the next  $n$  lines, you will be given the ages of these  $n$  family members. All ages will be non-negative numbers, with all ages on a separate line. Store these ages in an array called `ages`.

`ages[0]` will be Mr C's own age. Also, if a person's age is at index  $i$  in the array, then that person's mother's age will be at index  $2*i + 1$  and that person's father's age will be at index  $2*i + 2$ . In particular, Mr.C's mother's and father's ages are given at indices 1 and 2, respectively.

Your job is to check if the given ages are valid: that for every person, both their parents should be strictly older than the person since it is obviously an error if one's parents are the same age as oneself or are younger than oneself. If the given ages are valid, print the word "VALID" as the output. Otherwise, print the word "INVALID" in the output followed by a space followed by the number of parent child pairs which were in violation of the above age rule.

**Hint:** Although this question can be solved using loops, there are cute solutions that use recursion. Try a recursive solution to test your recursion prowess.

**Caution**

1. Array indexing in C starts from 0
  2. Be careful not to double count parent-child pairs in case the ages are invalid
  3. Count only invalid parent-child pairs, not grandparent-grandchild pairs, etc.
  4. Be careful about extra/missing lines and extra/missing spaces in your output.
- 

**EXAMPLE 1:**

INPUT

```
3
1
2
3
```

OUTPUT:

VALID

**Explanation:** Mr.C has age 1, and his parents have ages 2, 3 and are thus strictly older than him.

**EXAMPLE 2:**

INPUT

```
3
2
1
2
```

OUTPUT:  
INVALID 2

**Explanation:** Mr C is older than his mother and is of the same age as his father, both relationships are in violation of the age rule since children must be strictly younger than their parents.

-----

**Grading Scheme:**

Total marks: **[20 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 points. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 1 point and each hidden test case is worth 2 points. There are 2 visible and 4 hidden test cases.

 **Start Solving!** (</editor/practice/6231>)