GECG10069(561085) F25

Final Examination

FULL SCORES:

50 %

EXAMINATION TIME:

9：00~12：00，total 180 minutes

INSTRUCTIONS:

- You may pick an arbitrary number of problems to solve with a total score of up to 50%.
- Please strictly follow the input and output specifications described in each problem. Any deviation that causes test case errors will result in point deductions, for which the student is fully responsible.
- Name your files by problem number: Problem 1 → p1.cpp, …
- Zip all program files into one archive named with your student ID (e.g., 411511004.zip or 411511004.7z). Submit through **NTU Cool**. If the system is slow, other methods will be announced.
- You have a 10-minute grace period (until 12:10 PM) to submit.
- **Submissions after 12:10 PM will not be accepted.**
- You are allowed to open any notes or books. However, you are **not allowed to copy others' code AND browse on the internet**. Otherwise, you'll get 0% in this examination and be punished by the school regulations.
- Carefully read the statements and requirements of each problem.

PROBLEM 01 (10%)

You are a Pokémon Trainer and you want to write a simple program to determine whether you can successfully catch a Pokémon.

Each Pokémon has the following attributes:

- **Name** (string)
- **Rarity** (integer, 1–5) — a higher value means the Pokémon is harder to catch
- **CatchRate** (double, e.g., 0.8, 1.2, etc.) — a higher value increases the chance of catching

You are also given a **ballPower** (1–5), representing the strength of the Poké Ball you throw.

A capture is successful if:

- catchScore ≥ rarity ,and catchScore = ballPower * catchRate

Otherwise, the Pokémon escapes.

Input Format :

- The first line contains an integer N, the number of Pokémon.
- The next N lines each contain:

  <name(string)> <rarity(int)> <catchRate(double)>

- The final two lines contain:

  <ballPower(int)>

  <targetPokemonName(string)>

Output Format :

If the capture succeeds: Caught <name>

If the capture fails: <name> escaped

Constraints -

- *1 ≤ N ≤ 100*
- *Pokémon names contain no spaces and appear exactly as given in the input*
- *1 ≤ rarity ≤ 5*
- *catchRate > 0*
- *1 ≤ ballPower ≤ 5*

Sample Input - 1

```
3
Pikachu 2 1.5
Bulbasaur 3 2
Charmander 4 0.8
3
Pikachu
```

Sample Output - 1

```
Caught Pikachu
```

Sample Input - 2

```
3
Pikachu 2 1.5
Bulbasaur 3 2
Charmander 4 0.8
2
Pikachu
```

Sample Output - 2

```
Caught Pikachu
```

Sample Input - 3

```
3
Pikachu 2 1.5
Bulbasaur 3 2
Charmander 4 0.8
1
Pikachu
```

Sample Output - 3

```
Pikachu escaped
```

PROBLEM 02 (10%)

You are implementing a simple score processing system for a student database. Each record in the input describes either:

- A student's name
- A subject with a list of scores

You are recommended to read each input line using **getline**, and then use a **stringstream** to analyze its content.

There are two types of records in one line and these two types are in arbitrary order:

- **Input format**

1. Name Record:

   NAME <studentName>

   This sets the active student. All following score records belong to this student until another name record appears.

2. Score Record:

   SCORE <subjectName> <score1> <score2> ... <scoreN>

   - <subjectName> is a single word (no spaces)
   - <score1> ... <scoreN> are values intended to be integers
   - Negative integer number is valid
   - Scores may include invalid tokens (e.g., A+, x, 9a……). Only values that can successfully be parsed as integers should be counted.

Your task is to compute the **average score** for each valid SCORE record and output the result as a **floating-point number**.

- **Output format**

  For each SCORE record, output:

  <studentName> <subjectName> <averageScore>

The averageScore must be printed as a floating-point value.

Constraints –

- Student names contain no spaces
- Subject names contain no spaces
- **Invalid score tokens must be ignored**

Sample Input - 1

```
NAME Alice
SCORE Math 90 85 92 88
SCORE English 100 87
```

Sample Output - 1

```
Alice Math 88.75
Alice English 93.5
```

Sample Input - 2

```
NAME Bob
SCORE History 70 x 80 error 75
```

Sample Output - 2

```
Bob History 75
```

Sample Input - 3

```
NAME Carol
SCORE Science 100 A+ 90 85 9a
NAME David
SCORE Music 80 70
```

Sample Output - 3

```
Carol Science 91.6667
David Music 75
```

PROBLEM 03 (10%)

You are given N names (each without spaces), stored in a 2D C-string array **char names[50][31]**, and one keyword. Your task is to search for the first name that contains the keyword as a substring, **ignoring letter case**.

Print only the index **(0-based)** of **the first matched name**. If no name contains the keyword, print -1.

Input:

- The input contains several lines in the following format:
  - Line 1:
    N — number of names (*1 ≤ N ≤ 50*)
    M — maximum length of each name (*1 ≤ M ≤ 30*)
  - Line 2 ~ Line N+1
    <name1> … <nameN> (no spaces)
  - Line N+2
    keyword contains no spaces *(the length of the keyword ≤ 30)*

Output:

- If a match is found: **print its index (0-based)**
- Otherwise: **print -1**

Hint:

- You can use some **operations** (**strstr()**, etc.) from **<cstring>** and **tolower()** from **<cctype>** for case-insensitive comparison.

Sample Input - 1
5 20 Alice bobby Charlie alicia BOB Li

Sample Output - 1

```
0
```

Sample Input - 2

```
4 20
Zoe
Mark
Tina
Loki
Abc
```

Sample Output - 2

```
-1
```

Sample Input - 3

```
7 30
Zoe
alex
Bob
carol
Amy
Banana
Nanana
Na
```

Sample Output - 3

```
5
```

PROBLEM 04 (10%)

You are given an integer array "**arr**", of **fixed size 10**, and an integer N. Your task is to find two **different** indices i and j such that:

- $i < j$
- the value arr[i] + arr[j] is **as close as possible** to N

Formally, you want to minimize the absolute difference:

$$|(arr[i]+arr[j])-N|$$

Among all pairs (i, j) that achieve the **same minimal** absolute difference, choose the one with the **smallest i** , then the smallest j.

- You only need to output the indices i and j.

Input:

The input consists of **two lines**:

- **Line 1:** 10 integers, representing arr[0], arr[1], ..., arr[9], separated by single spaces.
- **Line 2:** one integer N.

Output:

Output **two integers** i and j (with $i < j$), separated by a space.
These indices **must satisfy**:

1. $i < j$,
2. | (arr[i] + arr[j]) - N | **is minimal** among all $0 \le i < j \le 9$,

Constraints:

• All elements of arr and N are **32-bit signed integers.** If tied, choose the pair **with the smallest i, then the smallest j.**

Sample Input - 1
5 1 9 3 7 8 2 6 4 10 11
Sample Output - 1
0 7

Sample Input - 2
2 2 4 0 1 -1 -1 4 -2 0 -2
Sample Output - 2
3 8
Sample Input - 3
-3 8 1 5 -2 7 4 0 -1 6 3
Sample Output - 3
0 9

PROBLEM 05 (10%)

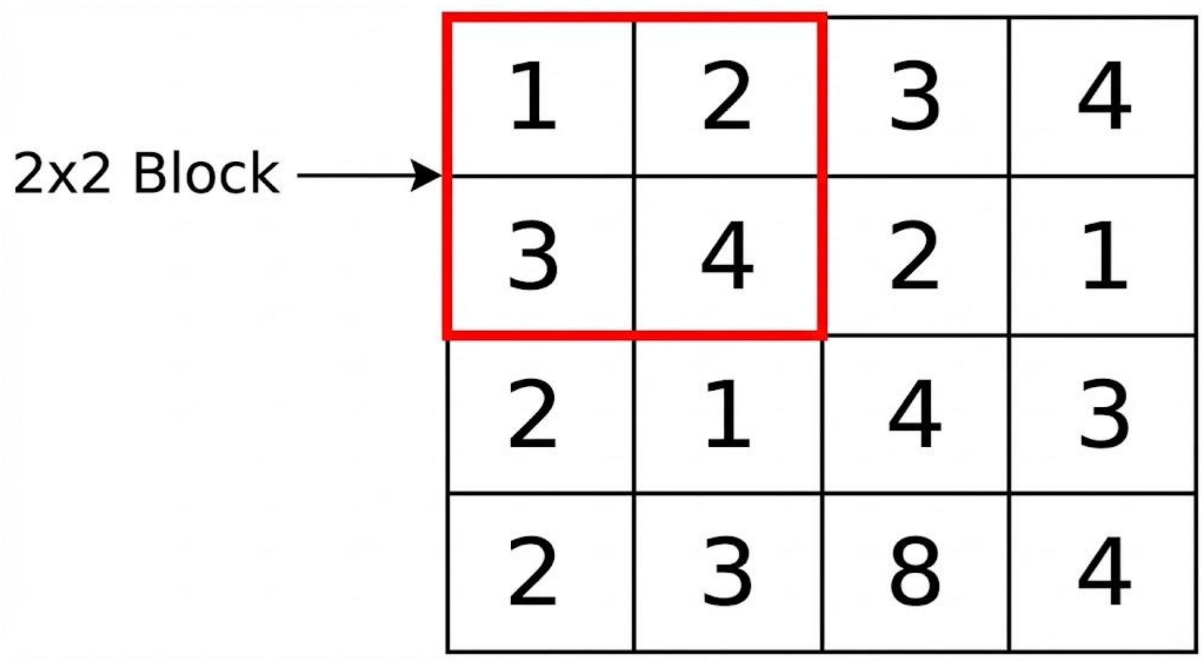
You are given a 4x4 grid representing a mini-Sudoku board. Your task is to determine if the grid is a valid solved Sudoku.

A 4x4 Sudoku is valid if and only if:

1. Each **row** contains the digits 1-4 exactly once.
2. Each **column** contains the digits 1-4 exactly once.
3. Each of the four **2x2 sub-blocks** contains the digits 1-4 exactly once.

The grid is divided into sub-blocks as follows:

- **Top-Left:** Rows 0-1, Cols 0-1
- **Top-Right:** Rows 0-1, Cols 2-3
- **Bottom-Left:** Rows 2-3, Cols 0-1
- **Bottom-Right:** Rows 2-3, Cols 2-3



Input:

The input consists of 4 lines. Each line contains 4 integers separated by single spaces, representing the rows of the grid.

Output:

Output a single line:

- Print **Valid** if the grid satisfies all Sudoku rules.
- Print **Invalid** if the grid violates any rule.

Sample Input - 1

1 2 3 4
3 4 1 2
2 1 4 3
4 3 2 1

Sample Output - 1
Valid
Sample Input - 2
1 2 3 4 1 2 3 4

```
3 4 1 2
4 3 2 1
```

Sample Output - 2

```
Invalid
```

Sample Input - 3

```
1 2 3 4
2 1 4 3
3 4 1 2
4 3 2 1
```

Sample Output - 3

```
Invalid
```