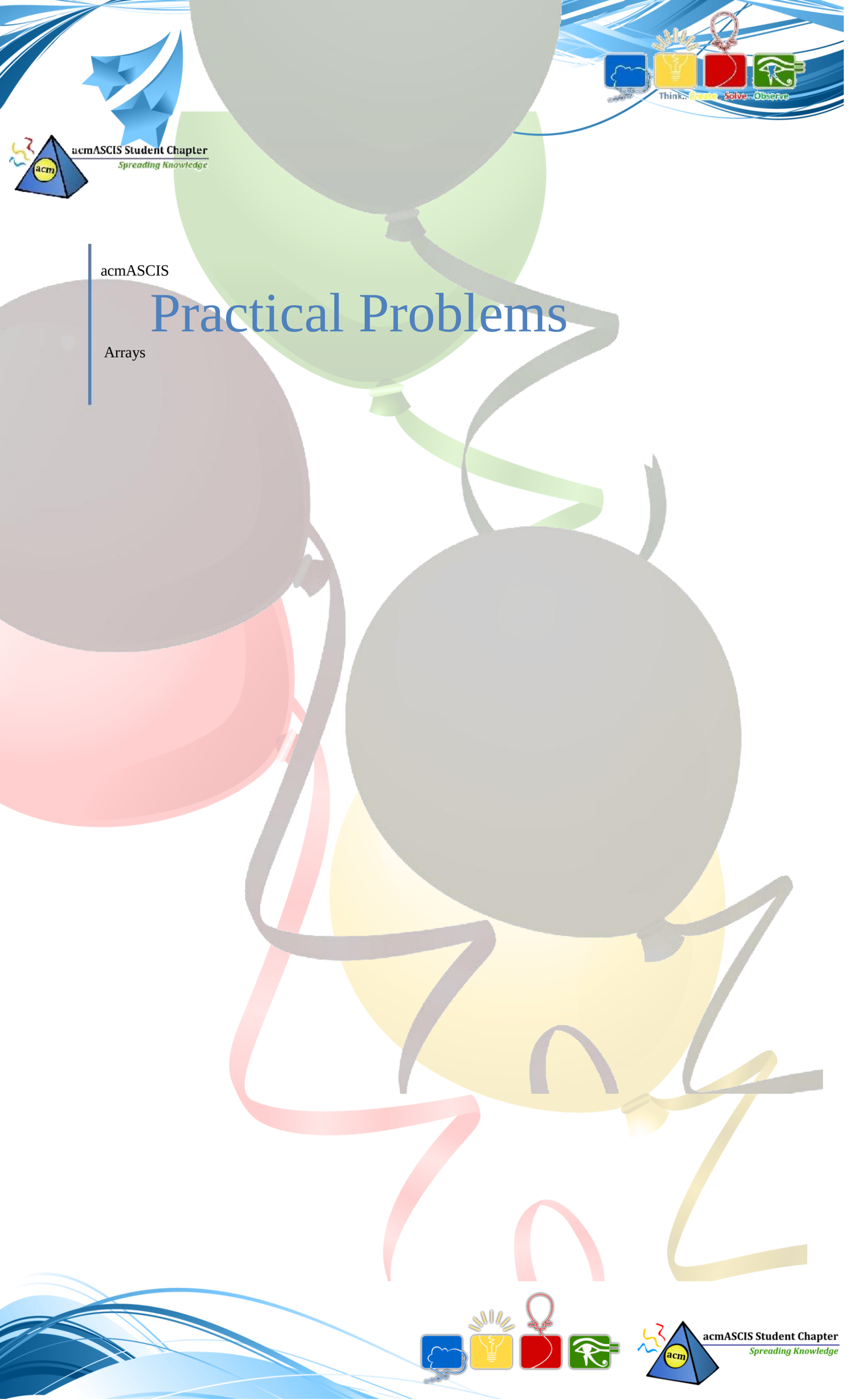


acmASCIS

Practical Problems

Arrays

Brick Game

There is a village in Bangladesh, where brick game is very popular. Brick game is a team game. Each team consists of odd number of players. Number of players must be greater than **1** but cannot be greater than **10**. Age of each player must be within **11** and **20**. No two players can have the same age. There is a captain for each team. The communication gap between two players depends on their age difference, i.e. the communication gap is larger if the age difference is larger. Hence they select the captain of a team in such a way so that the number of players in the team who are younger than that captain is equal to the number of players who are older than that captain.

Ages of all members of the team are provided. You have to determine the age of the captain.

Input

Input starts with an integer **T** (**T** ≤ **100**), the number of test cases.

Each of the next T lines will start with an integer **N** (**1** < **N** < **11**), number of team members followed by N space separated integers representing ages of all of the members of a team. Each of these N integers will be between **11** and **20** (inclusive). Note that, ages will be given in strictly increasing order or strictly decreasing order. We will not mention which one is increasing and which one is decreasing, you have to be careful enough to handle both situations.

Output

For each test case, output one line in the format **"Case x: a"** (quotes for clarity), where **x** is the case number and **a** is the age of the captain.

Sample Input:

```
2
5  19  17  16  14  12
5  12  14  16  17  18
```

Sample Output:

```
Case 1: 16
Case 2: 16
```

SMS Typing

Cell phones have become an essential part of modern life. In addition to making voice calls, cell phones can be used to send text messages, which are known as SMS for short. Unlike computer keyboards, most cell phones have limited number of keys. To accommodate all alphabets, letters are compacted into single key. Therefore, to type certain characters, a key must be repeatedly pressed until that character is shown on the display panel.

	abc	def
ghi	jkl	mno
pqrs	tuv	wxyz
	<SP>	

In the above grid each cell represents one key. Here **SP** means a space. In order to type the letter **'a'**, we must press that key once, however to type **'b'** the same key must be repeatedly pressed twice and for **'c'** three times. In the same manner, one key press for **'d'**, two for **'e'** and three for **'f'**. This is also applicable for the remaining keys and letters. Note that it takes a single press to type a space.

Input

The first line of input will be a positive integer **T** where **T** denotes the number of test cases. **T** lines will then follow each containing only spaces and lower case letters. Each line will contain at least **1** and at most **100** characters.

Output

For every case of input there will be one line of output. It will first contain the case number followed by the number of key presses required to type the message of that case. Look at the sample output for exact formatting.

Sample Input:

```
2
welcome to ulab
good luck and have fun
```

Sample Output:

```
Case #1: 29
Case #2: 41
```

Above Average

It is said that 90% of frosh expect to be above average in their class. You are to provide a reality check.

Input

The first line of standard input contains an integer **C**, the number of test cases. **C** data sets follow. Each data set begins with an integer, **N**, the number of people in the class (**1** <= **N** <= **1000**). **N** integers follow, separated by spaces or newlines, each giving the final grade (an integer between **0** and **100**) of a student in the class.

Output

For each case you are to output a line giving the percentage of students whose grade is above average, rounded to **3 decimal places**.

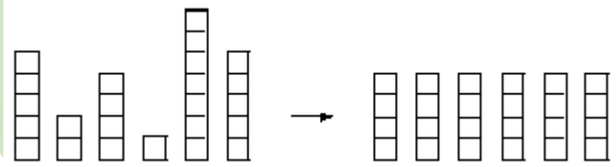
Sample Input:

```
5
5  50  50  70  80  100
7  100  95  90  80  70  60  50
3  70  90  80
3  70  90  81
9  100  99  98  97  96  95  94  93  91
```

Sample Output:

```
40.000%
57.143%
33.333%
66.667%
55.556%
```

Box Of Bricks



Little Bob likes playing with his box of bricks. He puts the bricks one upon another and builds stacks of different height. ``Look, I've built a wall!", he tells his older sister Alice. ``Nah, you should make all stacks the same height. Then you would have a real wall.", she retorts. After a little con- sideration, Bob sees that she is right. So he sets out to rearrange the bricks, one by one, such that all stacks are the same height afterwards. But since Bob is lazy he wants to do this with the minimum number of bricks moved. Can you help?

Input

The input consists of several data sets. Each set begins with a line containing the number **n** of stacks Bob has built. The next line contains **n** numbers, the heights $\mathbf{h_i}$ of the **n** stacks. You may assume $\mathbf{1 \leq n \leq 50}$ and $\mathbf{1 \leq h_i \leq 100}$.

The total number of bricks will be divisible by the number of stacks. Thus, it is always possible to rearrange the bricks such that all stacks have the same height.

The input is terminated by a set starting with **n** = **0**. This set should not be processed.

Output

For each set, first print the number of the set, as shown in the sample output. Then print the line **"The minimum number of moves is k."** , where **k** is the minimum number of bricks that have to be moved in order to make all the stacks the same height.

Output a blank line after each set.

Sample Input:

```
6
5 2 4 1 7 5
0
```

Sample Output:

```
Set #1
The minimum number of moves is 5.
```

Train Swapping

At an old railway station, you may still encounter one of the last remaining ``train swappers''. A train swapper is an employee of the railroad, whose sole job it is to rearrange the carriages of trains.
Once the carriages are arranged in the optimal order, all the train driver has to do, is drop the carriages off, one by one, at the stations for which the load is meant.

The title ``train swapper'' stems from the first person who performed this task, at a station close to a railway bridge. Instead of opening up vertically, the bridge rotated around a pillar in the center of the river. After rotating the bridge 90 degrees, boats could pass left or right

The first train swapper had discovered that the bridge could be operated with at most two carriages on it. By rotating the bridge 180 degrees, the carriages switched place, allowing him to rearrange the carriages (as a side effect, the carriages then faced the opposite direction, but train carriages can move either way, so who cares).

Now that almost all train swappers have died out, the railway company would like to automate their operation. Part of the program to be developed, is a routine which decides for a given train the least number of swaps of two adjacent carriages necessary to order the train. Your assignment is to create that routine

Input

The input contains on the first line the number of test cases (**N**). Each test case consists of two input lines. The first line of a test case contains an integer **L**, determining the length of the train ($\mathbf{0} \leq \mathbf{L} \leq \mathbf{50}$). The second line of a test case contains a permutation of the numbers 1 through **L**, indicating the current order of the carriages. The carriages should be ordered such that carriage 1 comes first, then 2, etc. with carriage **L** coming last.

Output

For each test case output the sentence: **"Optimal train swapping takes S swaps."** where **S** is an integer.

Sample Input:

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

Sample Output:

```
Optimal train swapping takes 1 swaps.
Optimal train swapping takes 6 swaps.
Optimal train swapping takes 1 swaps.
```