<<COVER PAGE>>

# Table of Contents

# A. 2048

2048 is a sliding block puzzle game. The game's objective is to slide numbered tiles on a grid to combine them to create a tile with the number 2048. It is played on a 4×4 grid, with numbered tiles that slide smoothly when the player moves them using the four arrow keys. Every turn, a new tile will randomly appear in an empty spot on the board with a value of either 2 or 4. Tiles slide as far as possible in the chosen direction until they are stopped by either another tile or the edge of the grid. If two tiles of the same number collide while moving, they will merge into a tile with the total value of the two tiles that collided. The resulting tile cannot merge with another tile again in the same move. The user's score starts at zero, and is incremented whenever two tiles combine, by the value of the new tile.
Write a program that takes the current user score, M number of moves and random appeared numbers (and their location on grid) and the current grid values as inputs and print the final score of the user.

**Input:**

First line contains current user score. 0 <= S

Next line contains M next moves. 0 <= M

- [move][2|4][at-row][at-column]

  eg: L200, Left arrow is pressed and then a new 2 is appeared at 0x0
- Move: L for left, U for up, R for right, D for down

Next line contains current grid content in a flatten string.

- From top left (0x0) to right bottom (3x3)
- Value can be 2,4,8,16,32,64,128,256,512, or 0 (zero indicate empty cell) separated by space.

**Output:**

Print the user score after given moves.

<u>**Check the example in next page.**</u>

**Example**

**Input:**

```
200
L213D202R211D210D200D212R410R200D401L212
0 2 0 0 0 4 0 0 2 8 2 0 8 32 16 2
```

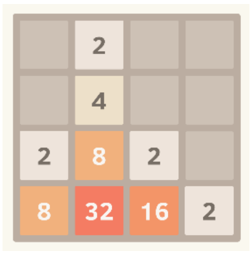
**Output:**

```
348
```

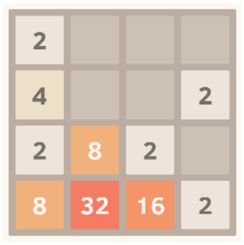
**Description:**

Note the user score is `200`

The below is 2048 grid for the input "`0 2 0 0 0 4 0 0 2 8 2 0 8 32 16 2`":



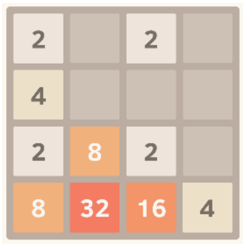
Input moves are: `L213,D202,R211,D210,D200,D212,R410,R200,D401,L212`



L213:

Left, a new 2 at 1x3

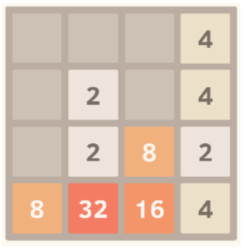
No merge



D202:

Down, a new 2 at 0x2

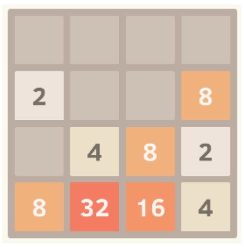
**4** at 3x3



R211:

Right, a new 2 at 1x1

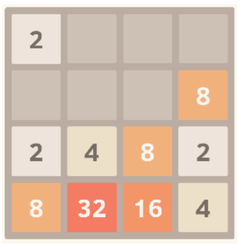
**4** at 0x3



D210:

Down, a new 2 at 1x0

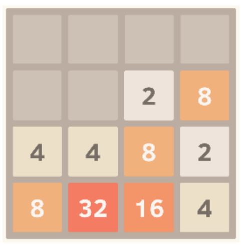
**8** at 1x3, 4 at 2x1



D200:

Down, a new 2 at 0x0

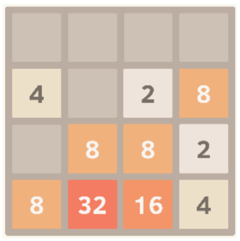
No merge



D212:

Down, a new 2 at 1x2

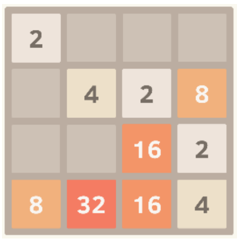
**4** at 2x0



R410:

Down, a new 4 at 1x0

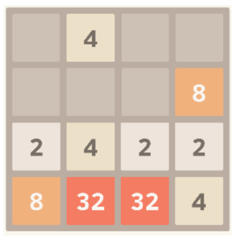
**8** at 2x1



R200:

Right, a new 2 at 0x0

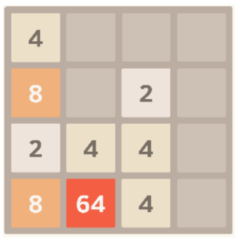
**16** at 2x2



D401:

Down, a new 4 at 0x1

**32** at 3x2



L212:

Left, a new 2 at 1x2

**4** at 2x2, **64** at 3x1

Add the merge scores together:

4 + 4 + 8 + 4 + 4 + 8 + 16 + 32 + 4 + 64 = 148

Add 148 to starting score: **200 + 148 = 348**

# B. TATCK

Last night Taliban attacked a police station. Afghan National Police fought them and repelled the attack by killing all of them. However, after the attack, Police forces noticed a strange pattern of tiny holes in the station's wall. They decide to cover these holes by a kind of bulletproof sheet metal temporarily. These sheets are only in square shape in different sizes, so they should use them to cover near to each other holes. Therefore, they first disjoint the holes to N parts based on their distance from each other. Then calculate and select appropriate sheet metal to cover each part.

Write a program to find the area of the smallest square that will cover all the holes in a part.

**Input:**

The first line contains a single integer N expressed in decimal without leading zeros, denoting the number of parts (Test Cases).
The subsequent lines describe the number of holes in each part and their location.

To better understand, each test case begins with a single line, containing a single integer N expressed in decimal without leading zeros, the number of holes to follow; each of the following N lines contains two integers x and y, both expressed in decimal without leading zeros, giving the coordinates of one of points:

- $N <= 30$
- $-500 < x, y < 500$

**Output:**

Print, on a single line with two decimal places of precision, the area of the smallest square containing all your points.
An answer will be accepted if it lies within 0.1 of the correct answer.

**Check the example in next page.**

**Example**

**Input:**

```
2
3
-1 -1
1 1
1 2
4
5 2
5 -2
-5 2
-5 -2
```

**Output:**

```
6.50
98.00
```

# C. PRIM

Given an array of integer numbers with 5 elements and for each of its elements, find the smallest prime number which is not less than that element.

**Input:**

The first line shows the number of test cases (T).

For each test case, we will have an array of integers (I) with length n in space-separated format.

$0 < I < 10^6$

$0 < n < 10^6$

**Output**:

Print the sum of obtained prime numbers for each test case.

**Example**

**Input:**

```
2
1254 12 1859 9762 6
5 6 8 10 15
```

**Output**:

```
12907
51
```

# D. LJUST

Nili company want to develop a simple text editor. Mohammad as the member of development team has been assigned to work on an essential feature - the justify line. You have already used a word processing software and specifically the justify line feature. Help him and write a program that gets the lines as input and print them out justified.

Here is what you should keep in mind while writing the program:

- Each line is a paragraph.
- To justify, pad the line by adding extra white spaces between words from left to right. Keep the white spaces balanced between words - Don't add all white spaces after the first word!!!
- Only justify if the number of characters in line is >= floor(C/2)
- Lines longer than C should be wrapped to next line.
- No justified line should start or ends with whitespace.
- Do not remove blank lines. Blank line is a line that does not have any non-blank character in it.

**Input:**

First line contains the max number of characters per line (C >= 50).

Next N lines (N < 100000) each containing W words (W < 10000).

Input ends with the line ENDOFINPUT. It is not part of input.

**Note:** There is no word bigger than C.

**Output:**

Print the lines justified.

**Check the example in next page.**

**Example**

**Input:**

```
50
The International Collegiate Programming Contest,
known as the ICPC, is an annual
multi-tiered competitive programming competition among the

universities of the world.
Headquartered at Baylor University,
directed by ICPC Executive Director and Baylor
Professor Dr. William
ENDOFINPUT
```

**Output:**

```
The  International Collegiate Programming Contest,
Known     as    the    ICPC,    is    an    annual
multi-tiered  competitive  programming competition
among the

universities         of         the         world.
Headquartered      at      Baylor      University,
directed  by  ICPC  Executive  Director and Baylor
Professor Dr. William
```

**Description:**

There should be only 50 characters in a line. So:

- First line could not be longer than those five words (49 characters long - including spaces), and only one space was needed to make it justified which was added after first leftmost word.
- Second line was 32 characters long >= floor(50 / 2), so 19 more spaces were added to make it justified.
- The third line was justified and the extra words ("among the") were wrapped into next line - that cannot be justified.

# E. BNKQ

Nili bank provides banking services to its customers throughout the country. In each of their branches they have multiple tellers (cashier) that provide deposit or withdraw services to customers. Each teller has a queue of customers that are waiting for depositing or withdrawing money from their account through the teller. The bank records customer, the time they stand in queue (or processed), and teller information.

With the information in their hand, bank want to know which tellers were the busiest and what is their peak time frame during the day.

Write a program that receive the records for a single day as input and print the top three busiest tellers and their peak time frame.

**Input:**

First N lines (N < 10000) contains the records for a single day.

- Record format: TELLER<whitespace>CUSTOMER[<whitespace>TIME]
- TELLER, is the teller id
- CUSTOMER, is the customer id
- TIME, is the time (between 08:00:00AM - 04:00:00PM) customer stands in queue, or was processed by the teller
- First occurrence of record is for customer standing in queue,
  Second occurrence of the same record is for the time customer being processed
- [<whitespace>TIME], it is absent in the record if customer leave the queue without processing.
  <u>Input ends with the line ENDOFINPUT. It is not part of input.</u>

**Output:**

Print top 3 busiest tellers, total number of customer they have processed in a day and his/her peak time frames (in hours); each teller in one line.

- Format: TELLER<whitespace>PROCESSED-COUNT<whitespace>PEAK-TIMEFRAME
- Being a busy teller means processing more customers, not having more customers in queue.
- Print the first PEAK-TIMEFRAME (the one nearest to 08:00:00AM) for each teller, if there are multiple time frames with same peak weight.
- Sort tellers by name in ascending order if two or more teller have processed same number of customers.
  - TIMEFRAME, each time frame is 1 hour in length.
    Eg: These are 8 timeframes between 08:00AM - 04:00PM:
    08AM, 09AM, 10AM, ... 02PM, and 03PM
  - A time frame is, eg:
    - 08:00:00AM <= 08AM time frame < 09:00:00AM
    - 03:00:00PM <= 03PM time frame < 04:00:00PM

<u>**Check the example in next page.**</u>

**Example**

**Input:**

```
T01 CUST01 08:18:55AM
T01 CUST03 08:20:00AM
T01 CUST01 08:22:00AM
T01 CUST02 08:20:23AM
T01 CUST02 08:25:17AM
T01 CUST03 08:27:50AM
T01 CUST04 10:07:46AM
T01 CUST04 10:19:21AM
T02 CUST05 10:40:11AM
T02 CUST06 10:40:18AM
T02 CUST05 10:44:30AM
T02 CUST06 10:46:44AM
T03 CUST07 02:18:55PM
T03 CUST08 02:19:00PM
T03 CUST07 03:19:10PM
T03 CUST08 03:21:27PM
T03 CUST09 03:26:19PM
T03 CUST09 03:29:59PM
T04 CUST10 11:13:34AM
T04 CUST10
ENDOFINPUT
```

**Output:**

```
T01 4 08AM
T03 3 03PM
T02 2 10AM
```

**Description:**

There are four tellers, and ten customers in this case.

- Teller 01, processed 4 customers. Three of them were processed in 08AM timeframe, one of them processed in 10AM timeframe.
- Teller 02, processed 2 customers. Both of them were processed in 10AM timeframe.
- Teller 03, processed 3 customers. One of them processed in 02PM timeframe, and the two others were processed in 03PM timeframe.
- Teller 04, had one customer in queue but left the queue.

So, the teller with most processed customer is T01 with four customers, T03 is the second most busiest teller with three customers, and then T02 that processed two.

Customer 01 (CUST01) stands in the teller 01 (T01) queue at
[T01 CUST01 08:18:55AM], and was processed at [T01 CUST01 08:22:00AM]

# F. ASLRDR

Suppose an assembly line in a factory with N stations. In each station, workers do an activity on the product that might be the same as previous or next stations' activity. The order of these stations is not important but they should be ordered such that the product insert to line from one side (Left or Right) and exit from other side (Right or Left) without reverse movement in line. Your job is writing a program to reorder an existing assembly line so that it passed the mentioned rule. You may reorder the assembly line in several "station swapping" but you only allow swap two adjacent stations.

**Input:**

The first line of input gives n, the number of assembly lines (Test Cases).
For each test case, one line of input follows, containing a string of up to 100 letters or digits that are the name of stations.

**Output:**

Output consists of one line per test case. This line will contain the least possible number of swaps, or "Impossible" if it is not possible to reorder the stations for passing the rule.

For example, assume we have 3 actions that are named 2,a and D which currently are ordered in an assembly line as "2a2aD". To pass the mentioned rule it should reorder to "2aDa2" with 3 swaps as follows:

- swap "aD" to yield "2a2Da"
- swap "2D" to yield "2aD2a"
- swap "2a" to yield "2aDa2"

**Example**

**Input:**

```
2
aj2a3b3bbb
aAAj2a3jb3bbb
```

**Output:**

```
Impossible
27
```

# G. SECHT

Miraqa and Sorya wanted to have a secret chat. To this aim, they used a simple rule as follows:
“Use the upper case of the left of the correct key if it is a letter otherwise use the lower case of the correct key”

Write a program for decoding an encoded text. The encoded text is formed base on mentioned rule.

**Input:**

Input contains one line. Which may contain both upper and lower case letters.

**Output:**

Print the decoded text. Note that you should echo the spaces directly to output.

**Example**

**Input:**

```
NUEaqa QW QUKK ARaER RGW aRRaXJ aR BUBW ON
```

**Output:**

```
MIRaqa WE WILL START THE aTTaCK aT NINE PM
```

# H. PTOFSUG

Ahmad shah is a patrol officer in Kabul. Every night he receives a large network of trails connecting some Police stations and he has to patrol between them during the night. Ahmad shah wants to find the shortest route that travels along every trail at least once. Help him to find that.

**Input:**

The first line of input for each case contains two positive integers: n <= 15, the number of police stations, and m < 1000, the number of trails. For each trail, there is one subsequent line of input containing three positive integers: the first two, between 1 and n, indicating the police stations at the end points of the trail; the third indicates the length of the that trail. There may be more than one trail between any two stations; each different trail is given only once in the input; each trail can be travelled in either direction. It is possible to reach any trail from any other trail by visiting a sequence of police stations connected by trails. Ahmad shah's route may start at any police station, and must end at the same station. A single line containing 0 follows the last test case.

**Output:**

For each case, there should be one line of output giving the length of Ahmad shah's route.

## Example:

**Input:**

```
4 5
1 2 3
2 3 4
3 4 5
1 4 10
1 3 12
0
```

**Output:**

```
41
```

# I. COPSCH

Senobar is a student and works in copy shop center of university at noon. The copy shop center of university receives many orders from professors and students.

Every day she receives orders for tomorrow and registers the reception time, the amount of order in 1000 pages unit and the determined deadline by the customers for each order. Then, she will do them tomorrow from 12:00.

She really likes do the best for her customers but sometimes the amount of orders is over than the capacity of copy shop's machine. Therefore, she cannot satisfy determined deadline by her customers always and concerned about the best schedule for doing the orders so that, the maximum amount by which an order's completion time overshoots its deadline is minimized. Help her to schedule the orders. Note that the capacity of the machine's page container is 1000 pages, which copy them in 1 hour. Moreover, she does not have to do an order at a stretch. She may do a unit of order A (1000 pages), then do some units of other orders and after that comes back to the order A.

**Input:**

The first line contains the number of orders, $T$ .Each of the next $T$ lines contains two integers, D and U that illustrate: the deadline of each order (in pm), and the number of pages (in unit: 1000 pages) respectively.

**Constraints**

$1 \leq T \leq 10^5$

$1 \leq U_i \leq 10^3$

**Output:**

Output should be in $T$ lines, which contain the value of the maximum amount by which an order's completion time overshoots its deadline, when the first $i$ orders on your list are scheduled optimally. See the sample input for clarification.

**Check the example in next page.**

**Example**

**Input:**

```
5
2 2
1 1
4 3
10 1
2 1
```

**Output:**

```
0
1
2
2
3
```

**Description:**

The first order alone can be completed in 2 hour, and so you won't overshoot the deadline.

With the first two orders, the optimal schedule can be:

time 1: order 2

time 2: order 1

time 3: order 1

We've overshot order 1 by 1 hour, hence returning 1.

With the first three orders, the optimal schedule can be:

time 1: order 2

time 2: order 1

time 3: order 3

time 4: order 1

time 5: order 3

time 6: order 3

Order 1 has a deadline 2, and it finishes at time 4. So, it exceeds its deadline by 2. Order 2 has a deadline 1, and it finishes at time 1. So, it exceeds its deadline by 0. Order 3 has a deadline 4, and it finishes at time 6. So, it exceeds its deadline by 2.

Thus, the maximum time by which you overshoot a deadline is 2. No schedule can do better than this.

Similar calculation can be done for the case containing 5 orders.

# J. KBLTRNS

Kabul is a populated city. However, the public transportation has not grown at this stage. As a result, people at morning and evening have many problems to get to their destinations. The good things about this city is, you can reach to your destination using different routes. Every route has associated time, and price. One day, Toryalai has an urgent job to complete. He was in hurry and wanted to get to his destination as soon as possible. Your job is to help Toryalai to reach to his destination at minimum amount of time. Toryalai can go to his destination using many routes and routes can be formed by different intersections. Help him finding the best route to from provided routes his destination.
Note: Toryalai cannot switch among routes.

**Input:**
The first line is the number of test cases. Second line (R) contains number of routes Toryalai can use to reach destination. Then, R lines containing the routes associated times and prices. Starting from A as a start point and D as a destination. Maximum time Toryalai has is 12 hours and associated time for every route intersections has given to you in minutes.
Note:
There is no route exceeding the maximum time.
Every route reaches to destination.

**Output:**
The time as Toryalai select the route with minimum associated time

**Example:**
**Input:**
```
1
2
A B 2 10Af, B C 5 40Af, C D 20 60Af
A F 1 40Af, F C 1 50Af, C D 20 30Af
```

**Output:**
```
22
```