South African Computer Olympiad
Final Round 2012
Day 1

Overview

Author(s)			
Problem	**bruce**	**mining**	**swallow**
Source	bruce.java bruce.py bruce.c bruce.cpp bruce.pas	mining.java mining.py mining.c mining.cpp mining.pas	swallow.java swallow.py swallow.c swallow.cpp swallow.pas
Input file	stdin	stdin	stdin
Output file	stdout	stdout	stdout
Time limit	0.25 seconds	0.5 seconds	1 second
Memory limit	64MiB	64MiB	64MiB
Number of tests	10	10	10
Points per test	10	10	10
Detailed feedback	No	No	No
Total points	**100**	**100**	**100**

The maximum total score is 300 points.

Finding Bruce

Introduction

Bruce likes to wander around his garden while contemplating important and difficult problems. Unfortunately an emergency has arisen and the government needs to find Bruce immediately so that he can help them resolve it. They don't know where he is; they only know how long he has been walking. Fortunately, Bruce wanders very predictably.

Task

Bruce's garden is an infinite plane divided into a grid of squares. He always starts the first second of his walk in the lower left corner of the grid, $(1, 1)$, and the plane extends arbitrarily far up and to the right. Bruce then moves in a particular pattern, moving one square per second. He starts by moving up one square, then one square to the right, and then down one square. He then moves one square to the right, up *two* squares, and left *two* squares. This is followed by moving up one square, then right *three* squares, and down *three* squares, and so forth. The grid below shows the path he travels, with the number at each grid location being the time at which he is at that location.

⋮	⋮	⋮	⋮	⋮	
25	24	23	22	21	⋯
10	11	12	13	20	⋯
9	8	7	14	19	⋯
2	3	6	15	18	⋯
1	4	5	16	17	⋯

Given the number of seconds Bruce has been walking in his garden, the government would like you to tell them at exactly what position Bruce is on the grid so that they can pick him up.

Example

From the grid above you can see that, for example, after 6 seconds of walking Bruce is at position $(3, 2)$ and after 23 seconds he is at position $(2, 5)$.

Input (stdin)

The input consists of a single integer, T, representing the number of seconds that Bruce has been walking.

Sample input

```
6
```

Output (stdout)

Your output should consist of two integers, separated by a space, representing the x and y coordinates of Bruce's location at time T.

Sample output

```
3 2
```

Constraints

- $1 \leq T \leq 2^{60}$

Additionally, in 70% of the test cases:

- $1 \leq T \leq 2^{42}$

Additionally, in 40% of the test cases:

- $1 \leq T \leq 2^{20}$

Time limit

0.25 seconds. (Python: 2.5 seconds)

Scoring

A correct solution will score 100% while an incorrect solution will score 0%.

Mining

Introduction

Fred the Manic Storekeeper has decided to enter the mining industry. He is eyeing a large plot of land which he believes holds great riches below the surface. However, he cannot decide precisely where to dig the mine.

Task

Fred has partitioned the land into cells. For each cell, he has determined how much profit he can expect to make by mining at that position. Some cells are resource rich and so give a large profit, while others are mainly dirt and so would result in a net loss.

Given a grid representing the profits obtainable from mining each of the cells (in thousands of rands), determine the rectangular region that would yield the greatest total profit, were Fred to build his mine there.

Implementation Note: The `max` function in Python is very slow. If you are using Python and require this functionality, it may be necessary to implement the comparisons yourself to achieve top marks.

Example

Suppose Fred provides us with the following grid.

1	1	-4
4	-1	3
-1	3	0

By choosing the lower 2x3 rectangle, Fred can achieve a profit of $4 - 1 + 3 - 1 + 3 + 0 = 8$ or R8 000. This choice is the best possible.

Input (stdin)

Sample input

```
3 3
1 1 -4
4 -1 3
-1 3 0
```

Output (stdout)

Sample output

```
8
```

Constraints

- $1 \leq N, M \leq 190$
- Each grid entry lies between $-1\,500$ and $1\,500$
- At least one entry will be positive.

Additionally, in 30% of the test cases:

- $N, M \leq 20$

And in 60% of the test cases:

- $N, M \leq 50$

Time limit

0.5 seconds. (Python: 10 seconds)

Scoring

A correct solution will score 100%, whereas an incorrect solution will score 0%.

Swallow Messaging

Introduction

Bruce and Carl live in the same area and constantly send each other messages. However, recently the local internet exchange went down and both have lost internet access.

But all is not lost, as both have a large reserve of messenger swallows (African, specifically) and coconuts! Unfortunately, the length of messages one can write on a coconut is severely limited. Therefore, they wish to compress the messages they send.

Thankfully, foreseeing this eventuality, both Bruce and Carl have agreed on a compression algorithm (described below). Your task (other than carving the messages on the coconuts) is to come up with a program to optimally compress the messages.

Compression Algorithm

The messages you will be compressing consist purely of uppercase letters (from A to Z). The compression algorithm works by encoding repeated subsequences as a number (indicating the number of repetitions) and the subsequence. For instance, ´TESTTESTTESTTEST´ would get encoded as '4(TEST)'.

But due to the notation we have chosen, we can recursively encode subsequences. For instance, `CATDOGDOGCATDOGDOG` could be compressed as `2(CAT2(DOG))`.

More formally, the decompression algorithm is, where S is a compressed string:

- If S consists of a single uppercase letter, then it decompresses to S.
- If S can be split into two compressed strings T and Q, which decompress to T' and Q' respectively, then S decompresses to $T'Q'$ (T' followed by Q').
- If S is a compressed string in the form $k(Q)$, where k is a decimal encoded integer and Q is a compressed string, then S decompresses to k repetitions of Q', where Q' is the decompression of Q.

Task

Given N and a message of length N, you have to output the shortest possible encoding of the message in the compression format.

If there are multiple possible ways of encoding the message you can output any one of them.

Example

Input (stdin)

The first line contains a single integer, N. The next line contains N uppercase letters.

Sample input

```
18
CATDOGDOGCATDOGDOG
```

Output (stdout)

A single line containing the shortest compression.

Sample output

```
2(CATDOGDOG)
```

Constraints

- $1 \leq N \leq 100$

Additionally, in 50% of the test cases:

- $1 \leq N \leq 13$

Time limit

1 second.

Scoring

A correct solution will score 100% while an incorrect solution will score 0%.