问题描述

Markdown 是一种很流行的轻量级标记语言（lightweight markup language），广泛用于撰写带格式的文档。例如以下这段文本就是用 Markdown 的语法写成的：

```
# Heading

## Sub-heading

Paragraphs are separated
by a blank line.

Text attributes _italic_.

Bullet list:

* apples
* oranges
* pears

A [link](http://example.com).
```

#

这些用 Markdown 写成的文本，尽管本身是纯文本格式，然而读者可以很容易地看出它的文档结构。同时，还有很多工具可以自动把 Markdown 文本转换成 HTML 甚至 Word、PDF 等格式，取得更好的排版效果。例如上面这段文本通过转化得到的 HTML 代码如下所示：

```
<h1>Heading</h1>

<h2>Sub-heading</h2>

<p>Paragraphs are separated
by a blank line.</p>

<p>Text attributes <em>italic</em>.</p>

<p>Bullet list:</p>

<ul>
<li>apples</li>
<li>oranges</li>
<li>pears</li>
</ul>

<p>A <a href="http://example.com">link</a>.</p>
```

#

本题要求由你来编写一个 Markdown 的转换工具，完成 Markdown 文本到 HTML 代码的转换工作。简化起见，本题定义的 Markdown 语法规则和转换规则描述如下：

●**区块**：区块是文档的顶级结构。本题的 Markdown 语法有 3 种区块格式。在输入中，相邻两个区块之间用一个或多个空行分隔。**输出时删除所有分隔区块的空行。**

○段落：一般情况下，连续多行输入构成一个段落。段落的转换规则是在段落的第一行行首插入 `<p>`，在最后一行行末插入 `</p>`。

○标题：每个标题区块只有一行，由若干个 `#` 开头，接着一个或多个空格，然后是标题内容，直到行末。`#` 的个数决定了标题的等级。转换时，`# Heading` 转换为 `<h1>Heading</h1>`，`## Heading` 转换为 `<h2>Heading</h2>`，以此类推。标题等级最深为 6。

○无序列表：无序列表由若干行组成，每行由 `*` 开头，接着一个或多个空格，然后是列表项目的文字，直到行末。转换时，在最开始插入一行 `<ul>`，最后插入一行 `</ul>`；对于每行，`* Item` 转换为 `<li>Item</li>`。本题中的无序列表只有一层，不会出现缩进的情况。

●行内：对于区块中的内容，有以下两种行内结构。

○强调：`_Text_` 转换为 `<em>Text</em>`。强调不会出现嵌套，每行中 `_` 的个数一定是偶数，且不会连续相邻。注意 `_Text_` 的前后不一定是空格字符。

○超级链接：`[Text](Link)` 转换为 `<a href="Link">Text</a>`。超级链接和强调可以相互嵌套，但每种格式不会超过一层。

#

输入格式

输入由若干行组成，表示一个用本题规定的 Markdown 语法撰写的文档。

输出格式

输出由若干行组成，表示输入的 Markdown 文档转换成产生的 HTML 代码。

样例输入

```
# Hello

Hello, world!
```

样例输出

```
<h1>Hello</h1>
<p>Hello, world!</p>
```

#

评测用例规模与约定

本题的测试点满足以下条件：

●本题每个测试点的输入数据所包含的行数都不超过100，每行字符的个数（包括行末换行符）都不超过100。

●除了换行符之外，所有字符都是 ASCII 码 32 至 126 的可打印字符。

●每行行首和行末都不会出现空格字符。

●输入数据除了 Markdown 语法所需，内容中不会出现 `#`、`*`、`_`、`[`、`]`、`(`、`)`、`<`、`>`、`&` 这些字符。

●所有测试点均符合题目所规定的 Markdown 语法，你的程序不需要考虑语法错误的情况。

每个测试点包含的语法规则如下表所示，其中“√”表示包含，“×”表示不包含。

测试点编号	段落	标题	无序列表	强调	超级链接
1	√	×	×	×	×
2	√	√	×	×	×
3	√	×	√	×	×
4	√	×	×	√	×
5	√	×	×	×	√
6	√	√	√	×	×
7	√	×	×	√	√
8	√	√	×	√	×
9	√	×	√	×	√
10	√	√	√	√	√

#

提示

由于本题要将输入数据当做一个文本文件来处理，要逐行读取直到文件结束，C/C++、Java 语言的用户可以参考以下代码片段来读取输入内容。

C++ 语言

```cpp
#include <iostream>
#include <string>

int main() {
  std::string line;
  while (std::getline(std::cin, line)) {
    ...
  }
}
```

C 语言

```c
#include <stdio.h>

#define BUF_SZ 101

int main(void) {
  char buf[101];
  while (fgets(buf, BUF_SZ, stdin)) {
    /* Processing code here */
  }
  return 0;
}
```

#