sort

`sort` 是 UNIX 及其变体中一个非常经典的命令，可用于将输入排序。

> 有兴趣的同学可以看看《AT&T Archives: The UNIX Operating System》。

在本题中，`sort` 默认将输入以行为单位并按照ASCII 码序排序，例如：

```
$ sort << EOF
heredoc> 1
heredoc> 10
heredoc> 2
heredoc> EOF
1
10
2
```

> 以 `$` 开始的行是用户的输入，有兴趣的同学可以了解下这种约定。
>
> `<< EOF` 称为 `heredoc`，有兴趣的同学可以戳这里了解。

其中以 `heredoc>` 开始的行是待排序的行（除了 `heredoc> EOF` 这一行），最后三行是程序的输出，可以看到其中 `10` 排在 `2` 前面（因为默认按照 ASCII 码序排序）。

要让 `sort` 按照数值排序，可以使用 `-n` 选项：

```
$ sort -n << EOF
heredoc> 1
heredoc> 10
heredoc> 2
heredoc> EOF
1
2
10
```

此外，还有逆序排序的 `-r` 选项：

```
$ sort -r << EOF
heredoc> 1
heredoc> 10
heredoc> 2
heredoc> EOF
2
10
1
```

本题中增加了 `-i` 选项表示忽略大小写进行比较：

```
$ sort << EOF
heredoc> a
heredoc> b
heredoc> Z
heredoc> EOF
Z
a
b

$ sort -i << EOF
heredoc> a
heredoc> b
heredoc> Z
heredoc> EOF
a
b
Z
```

解释：在 ASCII 码中，所有的大写英文字母排在小写英文字母之前，因此在前一种情况中 `Z` 在最前，忽略大小写之后，`Z` 就按照字母表的顺序排在了 `a` 的后面。

本题中还增加了 `-d` 选项表示只比较字母、数字和空格而忽略其他字符，例如：

```
$ sort << EOF
heredoc> a-c
heredoc> abd
heredoc> EOF
a-c
abd

$ sort -d << EOF
heredoc> a-c
heredoc> abd
heredoc> EOF
abd
a-c
```

解释：在 ASCII 码中，`-` 排在英文字母的前面，因此 `a-c` 会排在 `abd` 之前，而加上 `-d` 选项之后，两个字符串在第二次比较时就会使用 `c` 跟 `b` 比较，这样 `a-c` 就排在了后面。

所有可能的选项如下：

- 无选项，用 `-` 表示
- `n`
- `i`
- `d`
- `r`

使用函数指针实现 `sort` 命令的上述功能。

输入

第一行是数组的长度 `N`，保证 `N` 可用 `std::size_t` 容纳（可以通过引入头文件 `<cstddef>` 使用此类型）。

从第二行开始的 `N` 行均为字符串（可能包含空格！），字符串仅包含 ASCII 字母、数字、空格和标点符号，如果字符串是一个合法的整数（数学意义上的），不会出现前导零（例如，`023`），也不会出现正号 `+`，且必然能够用 `int` 容纳。

第 `N + 2` 行是命令数 `C`.

从第 `N + 3` 行开始的 `C` 行是命令。

每行输入保证以 `\n` 结尾。

输出

输出 `C` 组，每组 `N` 行，分别对应于相应的命令对数组进行排序的结果。

注意：在忽略大小写的情况下，对于英文意义上的同一个字母（例如，`A` 和 `a`），输出中大写字母应该排在小写字母之前。例如，输入为：

```
2
abc
ABC
1
i
```

输出应为：

```
ABC
abc
```

示例

示例 1

输入

```
3
1
10
2
2
-
n
```

输出

```
1
10
2
1
2
10
```

示例 2

输入

```
2
a-C
abd
3
-
d
i
```

输出

```
a-C
abd
a-C
abd
a-C
abd
```

提示

- 本题不允许使用除了 `std::string` 和 `std::vector` 之外的容器
- `std::string` 可以使用 `<` `>` `==` `!=` `<=` `>=` 进行比较
- 了解 C++ 标准库的 `sort` 函数
- 可以使用 `<cctype>` 头文件中的 `std::isalpha` `std::isdigit` `std::ispunct` 分别判断一个 `char` 是否是字母、数字和标点符号
- 可以使用 `std::getline` 读取一行数据

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

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

- 读取 `int` 之后，调用 `getline` 之前，需要使用 `std::cin >> std::ws` 跳过行中剩下的空白符

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

int main() {
    int n;
    std::cin >> n;
    std::cin >> std::ws;  // try to comment out the line
    std::string s;
    std::getline(std::cin, s);
}
```