Goal

Practice dynamic memory allocation in C by maintaining a list of books stored as a dynamic array of pointers to `Book`. You will implement exact-fit string allocation, removal by shifting and `realloc`, and full cleanup.

Data Type

Each book is represented by the following structure:

```
typedef struct {
    char *title;  // dynamically allocated string (exact size)
    int  pages;   // number of pages
} Book;
```

Program Behavior

1. Read Initial Books

a) Read an integer `n` — the number of books ($n \geq 0$).

b) For each book, read a single-word `title` and an integer `pages`:

```
<title> <pages>
```

Allocate `title` with exact size (`strlen(title) + 1`) and copy the contents.

c) Store the books in a dynamically allocated array of pointers:

```
Book **arr;   // length n
```

2. Process Queries

a) Read an integer `q` — the number of queries.

b) Each query is one of the following:

- **Type 1**: `1`
  Print all books, one per line:

  ```
  <title> <pages>\n
  ```

- **Type 2**: `2 <title>`
  Remove the book with the given title if present. Removal requires:
  i) free the removed book (`free(title)` then `free(Book)`),
  ii) shift elements to fill the gap,
  iii) shrink the array using `realloc` (or `free` and set to `NULL` if it becomes empty).

  This operation is *silent* (no output), even if the title is not found.

3. Cleanup

Free every `Book` (free `title`, then the `Book` struct) and finally free the array itself.

Input/Output Examples

Example Input

```
3
Dune 500
It 600
Emma 350
3
1
2 It
1
```

Example Output

```
Dune 500
It 600
Emma 350
Dune 500
Emma 350
```

Constraints and Notes

- Use only: `scanf`, `printf`, `malloc`, `realloc`, `free`, `strlen`, `strcpy`, `strcmp`.
- All titles are single tokens (no spaces).
- The array must be an array of pointers: `Book **`.
- Aim for no memory leaks: ensure all owned memory is freed before the program exits.