Phone book

Submission deadline:	**2023-12-11 11:59:59**	1352376.558 sec
Late submission with malus:	**2023-12-31 23:59:59** (Late submission malus: 100.0000 %)	
Evaluation:	**0.0000**	
Max. assessment:	**5.0000** (Without bonus points)	
Submissions:	0 / 20 Free retries + 10 Penalized retries (-10 % penalty each retry)	
Advices:	0 / 2 Advices for free + 2 Advices with a penalty (-10 % penalty each advice)	

The task is to develop a program that manipulates an electronic phone book.

Assume a simple phone book. The phone book keeps pairs phone number - name. The program further searches the phone book. The searches search either the phone number, or the name. Moreover, we do not have to specify complete phone/name, instead, we may only specify a prefix of the phone/name. The program searches the phone book and selects all records that match the given filter.

We assume the searches are in the form of a numbers. The numbers either denote (a prefix of) the phone number, or (a prefix of) a name in the T9 notation. That is, a search for name `Vagner` will look like `824637`.

The program is interactive. It processes individual input lines until it reaches end-of-file (EOF) in the standard input. Each line represents one command. The commands are:

- `+ phone name` This command adds a new record to the phone book. The number is a sequence of digits 0 to 9, the length of the phone is between 1 and 20 digits. Caution: leading zeroes are important. The number is any sequence of letters and spaces. The name cannot be empty, it must not start/end with a space. The character +, the phone and the name must be separated by exactly one space.
- `? number` This command searches the phone book. The number is any sequence of digits 0-9, the length is at least one character (there is not an explicit upper limit). The character ? and the number are separated by exactly one space.

The output consists of answers to the commands in the input. The add command may result in:

- an error it the format is invalid,
- an error if the command adds a record that already exists in the phone book (an exact match of phone and name, including matching upper/lower case letters),
- OK if the new record has been inserted to the phone book.

The search command may result in:

- an error if the format is invalid,
- the list of matching records and the number of matching records. This format will be used if there is at most 10 matching results,
- the number of matching results (i.e. the records are not printed). This format will be used if there is more than 10 matching results.

The program must validate input data. If the input is invalid, the program must detect it, it shall output an error message (see below) and skip the processing of that command (proceed to the next input line). If displayed, the error message must be displayed to the standard output (do not send it to the error output) and the error message must be terminated by a newline (\n). The input is considered invalid, if:

- an unknown command (command does not start with + or ?),
- phone or name is missing in the add command,
- number is missing in the search command,
- phone / number does not consist of digits,
- name does not consist of letters / spaces,
- name starts or ends with a space,
- missing/extra spaces separating commands/phone/name,
- phone in the add command is longer than 20 digits.

Invest some time to think up the structure of the program. There is not any explicit upper limit on the number of records/length of the name. Use dynamic memory allocation and structures.

The searching may be rather slow, especially if the number of records is high. The time limits are adjusted such that basic searching algorithm passes all tests except the bonus. An advanced algorithm is required to pass the bonus test.

Sample program runs:

```
+ 123456 Vagner Ladislav
OK
+ 987654321 Vanerka Jiri
OK
+ 824637 Vagner Jiri
OK
+ 8244278 Balik Miroslav
OK
+ 8243245 Vaclavik
OK
+ 192837 Taggart John
OK
+ 98244212 Vogel Josef
OK
? 824
123456 Vagner Ladislav
824637 Vagner Jiri
8244278 Balik Miroslav
8243245 Vaclavik
192837 Taggart John
Total: 5
? 82
123456 Vagner Ladislav
987654321 Vanerka Jiri
824637 Vagner Jiri
8244278 Balik Miroslav
8243245 Vaclavik
192837 Taggart John
Total: 6
? 37
Total: 0
+ 1000001 Vacatko
OK
+ 1000002 Vaclavek
OK
+ 1000003 Vaclavkova
OK
+ 1000006 Vagner Ladislav
OK
+ 1000007 Vacek
OK
+ 1000008 Vachek
OK
+ 1000009 Varga
OK
? 824
123456 Vagner Ladislav
824637 Vagner Jiri
8244278 Balik Miroslav
8243245 Vaclavik
192837 Taggart John
1000006 Vagner Ladislav
Total: 6
? 82
Total: 13
+ 123456 Novakova
```

```
OK
? 123
123456 Vagner Ladislav
123456 Novakova
Total: 2
? 123456
123456 Vagner Ladislav
123456 Novakova
Total: 2
? 1234567
Total: 0
? 10000
1000001 Vacatko
1000002 Vaclavek
1000003 Vaclavkova
1000006 Vagner Ladislav
1000007 Vacek
1000008 Vachek
1000009 Varga
Total: 7
+ 123456 Novakova Jana
OK
+ 234567 Novakova Jana
OK
+ 123456 Novakova Jana
Duplicate contact.
+ 123456 Novakova Jana
Duplicate contact.
+ 123456 novakova Jana
OK
? 123456
123456 Vagner Ladislav
123456 Novakova
123456 Novakova Jana
123456 novakova Jana
Total: 4
```

```
+ 123456 test
OK
+ 1234567 test
OK
+ 123456 testtest
OK
+ 123456 test
Duplicate contact.
+ 123456    test
Invalid input.
+ 123456789012345678901234567890 foo
Invalid input.
? test
Invalid input.
?
Invalid input.
test
Invalid input.
```

Advice:

- The listings above (sample runs) use regular font to display user inputs and bold font to display program output. This formatting exists only here, on the WWW page, to increase readibility of the listings. The actual program must output just the text without any markup, see the attached files.
- Do not forget the newline (\n) after the last output line.
- Do not use `int` data type to represent phone numbers.
- The program requires dynamic memory allocation.

- Use C libraries to solve the problem. Do not use C++ (STL).
- The program requires a reasonable implementation of the input processing. If your implementation needs to increase the size of an array, do not increase the size by just one element.
- It is recommended to read the individual input lines and process them as strings.
- **T9 codes** (both lower and upper case letters):

```
ABC    2
DEF    3
GHI    4
JKL    5
MNO    6
PQRS   7
TUV    8
WXYZ   9
space  1
```

- The order of the phones in the answer is not specified. The testing environment adjusts the order of the lines before it actually compares the results.
- A solution of this assignment may be used for code review. However, the program must pass all mandatory tests and all optional tests with 100% results to be eligible for code review (the program does not have to pass bonus test).

Sample data:		**Download**
Submit:	Browse... **No file selected.**	**Submit**

✓ **Reference**

- **Evaluator: computer**
  - Program compiled
  - Test 'Základní test s daty dle ukázky': success
    - result: 100.00 %, required: 100.00 %
    - Max. run time: 0.009 s (limit: 1.000 s)
    - Total run time: 0.016 s
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Test mezních hodnot': success
    - result: 100.00 %, required: 20.00 %
    - Max. run time: 0.064 s (limit: 2.000 s)
    - Total run time: 0.120 s
    - Optional test success, evaluation: 100.00 %
  - Test 'Test ošetření vstupních dat': success
    - result: 100.00 %, required: 20.00 %
    - Max. run time: 0.009 s (limit: 1.000 s)
    - Total run time: 0.170 s
    - Optional test success, evaluation: 100.00 %
  - Test 'Test náhodnými daty': success
    - result: 100.00 %, required: 20.00 %
    - Max. run time: 0.046 s (limit: 1.000 s)
    - Total run time: 0.207 s
    - Optional test success, evaluation: 100.00 %
  - Test 'Test náhodnými daty + mem debugger': success
    - result: 100.00 %, required: 20.00 %
    - Max. run time: 0.026 s (limit: 2.000 s)
    - Total run time: 0.142 s
    - Optional test success, evaluation: 100.00 %
  - Test 'Bonus - test rychlosti': success
    - result: 100.00 %, required: 100.00 %
    - Max. run time: 1.045 s (limit: 3.000 s)
    - Total run time: 1.045 s
    - Bonus test - success, evaluation: 125.00 %
  - Overall ratio: 125.00 % (= 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.25)
- Total percent: 125.00 %

- Early submission bonus: 0.50
- Total points: 1.25 * (5.00 + 0.50) = 6.88

		Total	Average	Maximum	Function name
SW metrics:	Functions:	**17**	--	--	--
	Lines of code:	**308**	**18.12 ± 10.58**	**43**	`parseLine`
	Cyclomatic complexity:	**74**	**4.35 ± 3.12**	**13**	`parseLine`