CS201c: Programming Evaluation 4
String algorithms: Tries
Due date: Thursday, November 14 by 11:59 pm

Problem Description:

Notation. For a string s, |s| denotes its length.

You have to implement a data structure which, at any point of time, maintains a collection S of distinct bit strings, where each bit string is of length **exactly 32 bits**.

Your data structure is initially empty. Further, it should support the following **four** operations:

insert(b : bit string of length 32):

Check if b is in set S.
If yes, return 0.
If no, add b to set S and return 1.

remove(b: bit string of length 32):

Check if b is in set S.
If no, return 0.
If yes, delete b from set S and return 1.

search1(p : (pattern) bit string of length at most 32):

Consider the set T(p, S) of all pairs (i, s) such that:

(i) 1<=i<=(32-|p|+1),
(ii) bit string s is currently in set S, and
(ii) the substring of s from location i to (i+|p|-1) matches pattern p.

The return value of the function is the cardinality |T(p,S)| of this set.

Note. The pattern p in the next search function, search2, can have one or more `?' (wildcard) characters. If a character in the pattern is `?' (the wildcard character), then it matches any alphabet in {0,1}. For example, let p="?0??10" and s="001111010", then the longest prefix of s that matches a prefix of p is "00111".

search2(p : a (pattern) string of length at most 32 over the alphabet {0, 1, ?}):

For a string s currently in set S, let L(s,p) denote the *maximum length prefix of s that matches a prefix of p*.

(Clearly, 0 <= L(s,p) <= |p|. For the example above, s="001111010", p="?0??10", and L(s,p)=5.)

Suppose set S currently has n (distinct) bit strings s_1, s_2, …, s_n. Then function search2 outputs a single integer equal to the value of the following sum:

L(s_1,p) + L(s_2, p) + … + L(s_n, p)

Requirements.

1. *Running time.* The maximum number of operations in the RAM model allowed for each of the above functions are:

Function	Maximum number of operations (in RAM model)
insert	4000
remove	4000
search1	4000
search2	(4000)*(1+L), where L is the integer output by search2.

Note that the above table gives the **maximum number** of RAM operations, there are no hidden multiplicative constants (due to order notation) in these bounds on the running times.

2. You cannot use any in-built libraries (including standard template library). All data structures should be implemented in C++ from scratch.

3. Collaboration is not permitted on this assignment. Your submitted code should be completely your own.

See section titled ``Honor Code" in course outline already shared with you.

Input and Output Format:

Input format. The first line contains a single integer m, which denotes the number of succeeding lines in the input.

After this, each line contains the specification for one function call.

The code for functions insert, remove, search1, and search2 are 1, 2, 3, and 4 respectively.

Function call	Input line format
Call insert with bit string b.	“1 b” (integer 1 and bit string b separated by single whitespace)
Call remove with bit string b.	“2 b” (integer 2 and bit string b separated by single whitespace)
Call search1 on pattern p (without wildcard characters)	“3 p” (integer 3 and pattern p separated by single whitespace)
Call search2 on pattern p (with wildcard characters)	“4 p” (integer 4 and pattern p separated by single whitespace)

Output format.

(i) There is no output for insert and remove function calls.
(ii) For each search1 function call, print its return value on a line by itself.
(iii) For each search2 function call, print its return value on a line by itself.

Example.

For simplicity, this example uses 8-bit strings. (The actual input will have 32-bit strings.)

Sample Input.

```
12
1 10110101
1 01010111
1 10101111
1 00001110
1 00010001
2 01100110
```

3 101
3 0101
3 000
4 1??1
4 0??1
4 ?????111

Sample Output.
7
4
4
7
11
34

Reason. After first five insert operations, S = {s1, s2, s3, s4, s5}, where s_i is the bit string added by the i-th insert operation. The remove operation is unsuccessful.

For p="101", T(p,S) = { (1,s1), (4, s1), (6, s1), (2, s2), (4, s2), (1, s3), (3, s3), ~~(3, s5)~~ }
Thus, next output line: 7

For p="0101", T(p,S) = { (5, s1), (1, s2), (3, s2), (2, s3) }
Thus, next output line: 4

For p="000", T(p,S) = { (1, s4), (2, s4), (1, s5), (5, s5) }
Thus, next output line: 4

For p="1??1":
L(s1,p)=4
L(s2,p)=0
L(s3,p)=3
L(s4,p)=0
L(s5,p)=0

Thus, next output line: 7

For p="0??1":
L(s1,p)=0
L(s2,p)=4
L(s3,p)=0
L(s4,p)=3
L(s5,p)=4

Thus, next output line: 11

For p="?????111":

L(s1,p)=6
L(s2,p)=8
L(s3,p)=8
L(s4,p)=7
L(s5,p)=5

Thus, next output line: 34