# Spring 2013 Programming Competition

Run by UPE at Rensselaer Polytechnic Institute

November 17, 2013
2:30pm - 4:30pm

Sponsored By



# 1 Longest Palindrome

A palindrome is a word where it and its reverse are equal. Given a sequence of characters, find the longest palindrome in the sequence.

The input will be first the length of the characters in the sequence and then the entire sequence on the next line.

You will return the length of the longest palindrome in the sequence.

## 1.1 Example

Given the following sequence of characters:

```
abaacdfghisarahpalindromeihefffehcefiqueeeuqifoobarchoo
```

The longest palindrome is

```
fiqueeeuqif
```

with length 11.

## 1.2 Single input

```
<length of character sequence>
<character sequence>
...
```

## 1.3 Limits

```
<number of test cases> = 200
10000 ≤ <length of character sequence> ≤ 30000
```

## 1.4 Single output

```
<length of longest palindrome in sequence>
```

## 1.5 Sample input

Note this is a smaller sample input

```
11
sffsnxrrxyn
```

## 1.6 Expected output

```
4
```

## 2 Prefix Search

You are given two lists of words $I$ and $Q$. For every word in $Q$ output the word in $I$ that shares the longest prefix. Break any ties by using the lexographically first solution.

The input will be first the length of $I$ and then each word in $I$ on separate lines. Then the length of $Q$ and each word in $Q$ on separate lines.

You are expected to output a list of words on separate lines that represents the solution to each word in $Q$.

### 2.1 Example

Given $I = \{$charizard, charmeleon, foobar, lemon$\}$ and $Q = \{$foo, charmander, orange$\}$. The solution list would be {foobar, charmeleon, charizard}. Note that the last match is charizard due to no prefix being match and charizard being the closest lexographic match.

### 2.2 Single input

```
<I length>
<first word in I>
<second word in I>
...
<Q length>
<first word in Q>
<second word in Q>
```

### 2.3 Limits

```
<number of test cases> = 200
75 ≤ <length of I> ≤ 150
75 ≤ <length of Q> ≤ 150
```

### 2.4 Single output

```
<solution to first word in Q>
<solution to second word in Q>
...
```

### 2.5 Sample input

Note this is a smaller sample input

```
7
adsum
```

```
apud
eneus
alia
absque
agri
aurum
6
alienus
aureus
advoco
Aldenard
acervus
adficio
```

### 2.6 Expected output

```
alia
aurum
adsum
absque
absque
adsum
```

# 3 Dungeon Hallway

You are an adventurer exploring a dungeon and you find large hallway with a lot of gold and a door to the outside at the end. The only problem is the only way to get across the hallway is to walk across some large tiles that shoot a set of arrows at you. After experimenting by tossing a few stones you know how many arrows get shot from each tile. Find the path through the hallway tiles that gets you shot with the least amount of arrows.

The input will be first the width of the hallway, and then the length of the hallway. Then the hallway will be input row by row with each cell in the row separated by spaces and each row on a new line.

You are expected to start from the top row and end somewhere on the bottom row. You can only move in the 8 cardinal directions from each tile, within the bounds of the hallway.

You are expected to return the total number of arrows you are hit with when you reach the gold.

## 3.1 Example

Say this is the input hallway in

```
1 2 3 4 5
3 1 4 1 6
2 7 1 8 2
1 6 1 8 0
```

Then the safest path is:

```
1
  1
    1
    1
```

And you would print 4 as the path length to your solution.

## 3.2 Single input

```
<width of hallway / number of tiles per row>
<height of hallway / number of rows>
<# arrows in row 1, tile 1> <# arrows in row 1, tile 2> ...
<# arrows in row 2, tile 1> <# arrows in row 2, tile 2> ...
...
```

### 3.3 Limits

<number of test cases> = 200
3 $\leq$ <width of hallway> $\leq$ 100
3 $\leq$ <height of hallway> $\leq$ 100
<# arrows in a tile> $\geq$ 0

### 3.4 Single output

<number of arrows taken through optimal path>

### 3.5 Sample input

```
3
3
1 14 2
27 26 15
2 11 48
```

### 3.6 Expected output

```
28
```

# 4 DNA Alignment

You are a scientist trying to figure out the ancestry of a animals. You have two sets of DNA gene samples, one from the orignal animals and another from the animals you're trying to identitfy. Your jobs is to match the DNA gene sample of the unknown animal with their closest ancestor with respect to their DNA alignments.

This means you have to find the ancestor that has the least amount of insertions, deletions, or in place mutations of its DNA sequence to match the unknown animal.

The input will be first the length of the set of orignal animals followed by their DNA gene samples on separate lines. Then it will be the length of the unknown set of animals followed by their DNA gene samples on separate lines.

You are expected to output the sequence of the anscestor that matches each respective unknown animal the closest. If there any ties, break them with the earlier animal in the list.

## 4.1 Example

Assume you have the following sequence of an original animal:

```
AATCGTACTACATGACTGATAATTTTTCAG
```

and the sequence from an unkown animal:

```
ATCGTACTACATTACTGATAATTTTTCAGA
```

You would do the following match:

```
(insert A)ATCGTACTACATT(change T into G)ACTGATAATTTTTCAGA(delete A)
AATCGTACTACATGACTGATAATTTTTCAG
```

and the number of changes needed would be 3 for the original sequence.

## 4.2 Single input

```
<number orignal sequences>
<original sequnce 1>
<original sequnce 2>
...
<number unknown sequences>
<unknown sequnce 1>
<unknown sequnce 2>
...
```

### 4.3 Limits

<number of test cases> = 200
50 $\leq$ <length of sequences> $\leq$ 100
20 $\leq$ <number of sequences> $\leq$ 50

### 4.4 Single output

<closest match to unknown sequence 1>
<closest match to unknown sequence 2>
...

### 4.5 Sample input

Note this is a smaller sample input

```
4
TCACATGGGACTC
TCACACAAAC
AGAGCACTTATTGAC
TATAACTTTAGTC
2
CACATGGGTAATC
AGAGCATTATATGAC
```

### 4.6 Expected output

```
TCACATGGGACTC
AGAGCACTTATTGAC
```