Due: Dec 10 (11:59pm).

Programming Assignment #2

CMSC 351 – Fall 2014

Rules

1) You may only use C/C++, Java.
2) Your program should use the standard input/output. For example C/C++ users should use scanf/printf/cin/cout and Java users should use system.in/system.out. For more information, you may read this: http://en.wikipedia.org/wiki/Standard_streams
3) Your program should not use more than 128MB of memory.
4) For each test case, your program may only run for 5 seconds. Therefore, you need to design efficient algorithms. The runtime requirements are explained in the problem statements.
5) You can only use the basic libraries for reading and storing data. You cannot use any of the sorted containers, linked lists, or algorithms in the standard libraries. You should implement them yourselves. For example, you can use Arrays, ArrayLists, Vectors. However, you may "not" use LinkedList, QList, Sets, Maps, Hashmaps, or sorting algorithms in the libraries.
6) All programs should be coded only by yourself. You may discuss problems with other students as long as there are no written notes. **All the source codes** submitted by the students will be checked with each other **automatically** using a software plagiarism detector. Please be aware of consequences of cheating, fabrication, facilitation, and plagiarism by reading the following website http://www.studentconduct.umd.edu.
7) The programming assignments are going to be graded by an automatic grader. The automatic grader is a program that compares the output of your program for a single test case against the correct output. As the automatic grader is a computer program, it is sensitive to the formatting, spelling, spacing, and etc. of your outputs. In fact, the presentation of your output has to be exactly the same as what is described in the problem statement. The grade for each test case of a problem is only 0 or 1, and there is no partial credit.
8) **Submitting:** For each problem there is a project in the submit server (submit.cs.umd.edu). You will submit only one file for each problem which is the source code. For each problem, the name of the file is specified in the problem statement.
9) If you need more info, please send me an email to mgholami@cs.umd.edu (Milad Gholami) or come to my office hours (Thursdays 2-4).

Problem 1: Palindromes (100 points)

Submit your program as: palindromes**.c,** palindromes**.cpp, or** palindromes**.java**

A palindrome is a sequence of one or more characters that reads the same from the left as it does from the right. For example, **Z**, **TOT,** and **MADAM** are palindromes, but **ADAM** is not.

Given a string *S* of *n* capital letters. How many ways can one remove a few symbols (maybe ***0***) that the rest of sequence becomes a palindrome. Variants that are only different by an order of removing should be considered the same.

Input Description

The input file contains several test cases (less than *10*). The first line contains an integer *T* that indicates how many test cases are to follow. Each of the *T* lines contains a string *S* with length at least *1* and at most *100*.

Output Description

For each test case output in a single line an integer – the number of ways.

Sample Input

```
3
BAOBAB
AAAA
ABA
```

Sample Output

```
22
15
5
```

Problem 2: Tree (150 points)

Submit your program as: tree.c, tree.cpp, or tree.java

Given a rooted tree with *n* vertices, numbered from 1 to *n* (vertex 1 is the root). Each vertex of the tree has a value. You should answer *q* queries. Each query is one of the following:

- Format of the query is "**find *v***". Let's write out the sequence of vertices along the path from the root to vertex v: $u_1, u_2, \ldots, u_k$ $(u_1 = 1;\ u_k = v)$. You need to find vertex u_i with the maximum value among the values of all vertices $u_1, u_2, \ldots, u_k$. If there are several vertices with the maximum value, pick the one with maximum value of i.
- Format of the query is "**change *v w***". You must change the value of vertex *v* to *w*.

Input Description

The first line contains two space-separated integers $n,\ q$ $(1 \leq n, q \leq 100{,}000)$. The second line contains *n* integers $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 1{,}000{,}000)$, where a_i represent the value of node *i*.

Each of the next $n - 1$ lines contains two integers x_i and y_i $(1 \leq x_i, y_i \leq n;\ x_i \neq y_i)$, denoting the edge of the tree between vertices x_i and y_i.

Each of the next *q* lines contains a query in the format that is given above. For each query the following inequalities hold: $1 \leq v \leq n$ and $1 \leq w \leq 1{,}000{,}000$.

Note that: there are no more than *100* queries that change the value of a vertex.

Output Description

For each query of the first type output the result of the query.

Sample Input

```
4 6
4 8 4 2
1 4
4 3
3 2
find 1
find 2
find 3
find 4
change 1 9
find 2
```

Sample Output

```
1
2
3
1
1
```

Problem 3: Increasing Shortest Path (Bonus: 200 points)

Submit your program as: path.c, path.cpp, or path.java

You are given a weighted directed graph of *n* nodes (the nodes are numbered from *1* to *N*), where the weights of the edges are distinct and positive. For each graph, you are also given a list of queries to answer.

Each query will be represented by 2 integers A B, which means you need to find the shortest path (the path with minimum sum of weights of its edges) which goes from node A to node B such that the weights of the edges in that path are in increasing order along the path, which means the weight of each edge in that path should be greater than the weight of the edge before it (unless it is the first edge in the path). Your task is to write a program which answers these queries.

Input Description

The first line contains *3* integers separated by a single space *N M Q* ($\mathbf{2} \leq \boldsymbol{N} \leq \mathbf{150}$), ($\mathbf{0} \leq \boldsymbol{M} \leq \mathbf{3,000}$) and ($\mathbf{1} \leq \boldsymbol{Q} \leq \mathbf{10}$) representing the number of nodes, the number of edges and the number of queries, respectively. Followed by *M* lines, each line contains *3* integers separated by a single space X Y Z ($\mathbf{1} \leq \boldsymbol{X}, \boldsymbol{Y} \leq \boldsymbol{N}$) ($\mathbf{1} \leq \boldsymbol{Z} \leq \mathbf{10,000}$) which represent a directed edge going from the node *X* to the node *Y* with cost *Z* (*X* and *Y* will be different). Followed by *Q* lines, each line contains 2 integers separated by a single space A B ($\mathbf{1} \leq \boldsymbol{A}, \boldsymbol{B} \leq \boldsymbol{N}$) which represent a query as described above (*A* and *B* will be different).

Output Description

For each test case, print a single line for each query which contains a single integer, the minimum sum of weights for a path between the given pair of nodes which satisfies the given constraints, or '-1' if there is no valid path between the given nodes which satisfies the given constraints.

Sample Input

```
6 6 3
1 2 7
1 3 3
3 2 5
2 4 4
2 5 6
2 6 9
1 4
1 5
1 6
```

Sample Output

```
-1
14
16
```