Indian Statistical Institute

Semester-I 2021-2022

M.Tech.(CS) - First Year

Lab Test 2 (20 January, 2022)

Subject: Computing Laboratory

Total: 60 marks Duration: 3 hrs.

SUBMISSION INSTRUCTIONS

1. Naming convention for your programs: `cs21xx-test2-progy.c` (assuming `cs21xx` denotes your roll number and `progy` denotes the program number).
2. To submit the solution files (`.c` or `.h`), ensure that they not password protected and mail them together to `<assignisik@gmail.com>` with the subject line as follows:
   `MTech (CS) 2021-23 cs21xx Computing Lab - labtest2`.
3. You may consult or use slides / programs provided to you as course material, or programs that you have written yourself as part of classwork / homework for this course, but please **do not** consult or use material from other Internet sources, your classmates, or anyone else.
4. Please make sure that your programs adhere strictly to the specified input and output format. **You may lose marks if your program violates the input and output requirements.**
5. Submissions from different students having significant match will be **debarred from evaluation**.

NOTE: Unless otherwise specified, all programs should take the required inputs from stdin, and print the desired outputs to stdout.

Q1. Suppose a one-way monorail route comprises n halts. In the said route, starting and ending points are demarcated as halt 0 and halt $n-1$, respectively. The costs of travel between every pair of halts are provided to you in real values. Find the minimal travel cost in the entire route between the starting and ending points. Note that, taking intermediate halts might save the cost of travel.

[20]

Input Format

The input (to be read from stdin) comprises the number of halts in the first line followed by a matrix denoting the travel cost between every pair of halts. The matrix elements are provided in a row-major fashion and are separated by spaces. An entry in the row i and column j of the matrix denotes the cost of travel in a path between the halts i and j. The travel costs are necessarily non-negative for the cases where $i > j$. For the rest of the cases, the travel costs are '-1', indicating the non-existence of path (because of one-way route) between a pair of halts.

Output Format

The output (to be printed to stdout) shows the minimal cost of travel between the starting and ending points.

Sample Input 0:

```
3
-1 2.5 5
-1 -1 2.5
-1 -1 -1
```

Sample Output 0:

```
5
```

Sample Input 1:

```
3
-1 10 15
-1 -1 10
-1 -1 -1
```

Sample Output 1:

```
15
```

Sample Input 2:

```
4
-1 10 20 30
-1 -1 5 10
-1 -1 -1 5
-1 -1 -1 -1
```

Sample Output 2:

```
20
```

Q2. Recall that the insertion and deletion happen at the rear and front of a queue, respectively. Let us define "leaky queue" as a special queue data structure from which the data items may automatically leak. There exists a *leak tolerance* level of every "leaky queue". If a "leaky queue" has a *leak tolerance* level of L then after every L insert or delete operations the data item in the middle (toward the front in case of a tie) will automatically leak from it. Note that, overflow (inserting into a full queue) and underflow (deleting from an empty queue) are valid insert and delete operations in this context.

Write a program that will take the maximum size and *leak tolerance* level of a "leaky queue", and insert or delete elements to show the data items in the queue.

Input Format

The first line of input (taken from stdin) is a pair of integers, namely the maximum size $SIZE$ of the "leaky queue" and the *leak tolerance* level L. It follows by a number of insert or delete operations stated line by line. An insert operation is stated as "$+ < data\_item >$". A delete operation is stated as "$-$".

Output Format
The output will print the list of data items (separated by single spaces) in the "leaky queue" after each insert or delete statement in the input line. The data items must be ordered from front to the rear of the queue. The data items are required to be printed even for the case of an overflow or underflow.

Sample Input 0

```
4 3
+ 1
+ 3
+ 5
+ 7
+ 9
+ 11
```

Sample Output 0

```
4 3
+ 1
1
+ 3
1 3
+ 5
1 5
+ 7
1 5 7
+ 9
1 5 7 9
+ 11
1 7 9
```

Sample Input 1

```
5 2
+ 3
-
```

```
-
+ 8
+ 5
```

Sample Output 1

```
5 2
+ 3
3
-

-

+ 8

+ 5
5
```

Sample Input 2

```
10 3
+ 2
+ 3
+ 9
-
```

Sample Output 2

```
10 3
+ 2
2
+ 3
2 3
+ 9
2 9
-
9
```

Q3. Given a set of n numbers, a strict ϕ-heavy hitter is defined to be a member of the set that has more than ϕ-fraction of occurrences, i.e., its count should be more than $\phi * n$ in the set. Write a program that takes a series of numbers and can efficiently return the strict $\frac{1}{2}$-heavy hitter of the set, if it at all exists.

[20]

Input Format
The input (to be read from stdin) is a series of numbers separated by spaces.

Output Format
The output (to be printed to stdout) will be the $\frac{1}{2}$-heavy hitter of the input series. If there exists no strict $\frac{1}{2}$-heavy hitter, return `NIL`.

Sample Input 0:

```
1 2 1 2 1 2 1 2
```

Sample Output 0:

```
NIL
```

Sample Input 1:

```
9 7 5 3 1 1 1 1 1
```

Sample Output 1:

```
1
```

Sample Input 2:

```
1.1 2.3 1.1 4.5 1.1 1.1 7.8
```

Sample Output 2:

```
1.1
```