CSCI 2100A / ESTR2102 Data Structures

Midterm Examination (Programming Part)
6:00 p.m. - 9:00 p.m., Friday, March 29, 2019

Room 924 Ho Sin Hang Engineering Building

Instructions

1. The programming midterm is an open-book and open-notes examination. You may bring what you can carry on printed (hard copy) materials. You MUST not take anything that can record program code electronically to the examination venue. You will not need a calculator for any calculation.
2. The operation system will be Ubuntu. The computer configuration will have these basic editors: vi/vim, gedit, Visual Studio Code and CodeBlocks (IDE). This system includes GDB for debugging.
3. You can use all the functions provided by standard C library as long as you've included the corresponding header file.
4. The examination will begin when the Chief TA starts the clock and will end when the Chief TA stops the clock, which is usually three hours after the starting time including any missing time due to technical or other difficulties.
5. The time limit for each problem is 1 second. The CPU can execute around 10^9 operations in 1 second.
6. If there are no special instructions, all input integers and output integers are in the range from -2^{31} to $2^{31} - 1$.
7. You are suggested to work on Problem A first and then others afterwards. The problems can be divided into three increasing difficult levels from the perspective of algorithm as judged by the instructors: [A], [B, C, D, E, F], and [G]. If you get stuck in one problem, try to solve the next one.
8. Anyone who attempts to spam the server either through excessive submissions, allocating large amount of unnecessary memory, etc. will be penalized severely.
9. Please switch your mobile phones to silent mode and place it under your seat, you are not allowed to use them during the exam.
10. If you want to go to the restroom, please ask the TAs for permission first.
11. If you leave early from the examination without informing the TAs, you will not be able to come back to the examination.

Problem A - Difference Between Sums of Odd and Even Numbers

Given a sequence of integers, return the difference between the sum of all odd numbers and the sum of all even numbers.

Input The first line is an integer N ($1 \leq N \leq 10^3$) indicating the number of test cases. Each test case contains two lines. The first line is an integer m ($1 \leq m \leq 200$) indicating the total number of integers in a sequence. The second line is a sequence of integers separated by spaces. There is one space between two integers and there is no space after the last integer.

Output The difference between sums of odd and even numbers.

Sample Input

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

Sample Output

```
-3
-7
8
-1
-8
```

Explanation There are 5 test cases.

```
(1 + 3 + 5) - (2 + 4 + 6) = 9 - 12 = -3
(-3) - (4) = -3 - 4 = -7
(1 + 7) - (8 + (-2) + (-6)) = 8 - 0 = 8
((-3) + (-1) + (-5) + 1 + 7) - 0 = -1
0 - (2 + 2 + 2 + 2) = 0 - 8 = -8
```

Problem B - Write Program for a Music Player

You are requested to write a program to output songs for a simple music player. Given a play list of length m with the initial song labels as $1, 2, 3, ..., m$, the music player allows two operations - `1:PLAY` and `2:REMOVE` (a song from the play list).

With the `PLAY` operation, the player plays the currently first song in the play list and then places it to the end of the play list after it's being played. With the `REMOVE` operation, the player removes the currently first song from the play list and never plays it again.

Input The first line is an integer N ($1 \leq N \leq 10^3$) indicating the number of test cases. Each test case contains two lines. The first line has two integers m and n, where m ($1 \leq m \leq 10^3$) denotes the number of songs in a play list and n ($0 \leq n \leq 10^4$) denotes the number of operations. Assume that the initial song list is labelled as $1, 2, 3, ..., m$. The second line is a sequence of operations (1: `PLAY`; 2: `REMOVE`) separated by spaces and there is no space after the last operation. Besides, you can assume that there is always at least one song remaining in the list after all operations.

Output Output the labels of the songs being played in one line. The song labels should be separated by spaces. Output 0 after all operations. There is no space after 0.

Sample Input

```
2
5 10
1 1 1 1 1 1 1 1 1 1
4 6
1 1 2 1 1 1
```

Sample Output

```
1 2 3 4 5 1 2 3 4 5 0
1 2 4 1 2 0
```

Explanation There are 2 test cases.

For the first test case, the 5 songs $[1, 2, 3, 4, 5]$ are played in a loop twice.

For the second test case, the initial songs in the play list are [1, 2, 3, 4].

1. After the first `PLAY`, the song 1 is played and the play list becomes [2, 3, 4, 1].
2. After the second `PLAY`, the song 2 is played and the play list becomes [3, 4, 1, 2].
3. Then a `REMOVE` operation is carried out, the currently first song in the play list - song 3 is removed. The play list becomes [4, 1, 2].
4. The rest three `PLAY` operations will play the song 4, 1, and 2, respectively.

Problem C - Evaluate Prefix Expressions

In prefix expressions, operators precede their operands. For example, "/ 2 1" in prefix notation corresponds to "2 / 1" in normal infix notation.

This problem asks you to write a prefix expression calculator to read a string containing a prefix expression, evaluate it, and print out the final computation result. The calculator should support **four** operators, $+$, $-$, $*$ and $/$, namely addition, subtraction, multiplication and division.

For your convenience, there are a few important points/assumptions you need to notice.

1. Division operations (i.e., '/') follow the behaviour of integer division in C language. No special consideration on rounding is needed. When there is division by zero, output the error message '*division by zero*'.

2. All operands are integers. All operators are integer operators. The intermediate and final computation results are all integers (except for the '*division by zero*' case).

Input The first line is an integer N ($1 \leq N \leq 10^3$), indicating the number of test cases. Each test case contains two lines. The first line is an integer m ($1 \leq m \leq 200$), the total number of operands and operators. The second line is a valid prefix expression, with operands and operators separated by spaces. There is a space between two tokens and there is no space after the last token. Each input operands is a digit (i.e., one of 0-9). You can assume all the input prefix expressions are valid.

Output The computation result of each prefix expression test case. If you detect that a divisor equals zero, then you need to output '*division by zero*'.

Sample Input

```
5
5
* 3 + 1 2
5
- 1 * 9 8
5
/ 1 - 2 2
7
+ 1 / 8 - 1 9
3
/ 7 2
```

Sample Output

```
9
-71
division by zero
0
3
```

Explanation There are 5 test cases.

```
* 3 + 1 2 is equivalent to 3 * (1 + 2)
- 1 * 9 8 is equivalent to 1 - 9 * 8
/ 1 - 2 2 is equivalent to 1 / (2 - 2)
+ 1 / 8 - 1 9 is equivalent to 1 + 8 / (1 - 9)
/ 7 2 is equivalent to 7 / 2
```

Problem D - kth Smallest Element in a Sorted Matrix

Given an $n \times n$ matrix, where each of the rows and columns are sorted in ascending order, find the kth ($k \leq n \times n$) smallest element in the matrix. For example:

```
matrix =                    matrix =
[[ 1,  5, 10],              [[ 1 ]],
 [ 5, 10, 13],
 [10, 13, 15]],
k = 8,                      k = 1,
return 13.                  return 1.
```

Input The first line is an integer N ($1 \leq N \leq 10^3$), indicating the number of test cases.

Each test case contains $n+1$ lines. The first line contains two integers, n ($1 \leq n \leq 10^2$) and k ($1 \leq k \leq n^2$) , which are separated by a space. The following n lines are the row vectors of the matrix, which are separated by a space and are sorted in an ascending order. There is no extra space at the end of the line.

Output For each test case, output the kth smallest element in the matrix in one line.

Sample Input

```
2
3 2
1 5 12
10 11 13
12 13 15
3 8
1 5 12
10 11 13
12 13 15
```

Sample Output

```
5
13
```

Explanation There are 2 test cases.

For the first test case, k=2, the 2nd smallest element is 5.

For the second test case, k=8, the 8th smallest element is 13.

Source LeetCode 378

Problem E - Longest Substring Without Repeating Characters

Given a string, find the length of the longest substring without repeating characters. Note that a substring is a continuous subspan of the string.

Input The first line is an integer N ($1 \leq N \leq 10^2$), indicating the number of test cases. From the second line, each line is a string as a test case. The string length without including the newline character of each test case is between 1 and 10^4. The input characters are selected from ASCII code 33 (decimal) to 126 (decimal). You can refer to Figure 1 in the last page for more details. No whitespace characters are included except that the last character of each string is a newline character.

Output The length of the longest substring without repeating characters.

Sample Input

```
3
ab+db+bb
####
!wwkew
```

Sample Output

```
4
1
3
```

Explanation There are 3 test cases.

For "ab+db+bb", the longest substring is "ab+d" with the length of 4.

For "####", the longest substring is '#'.

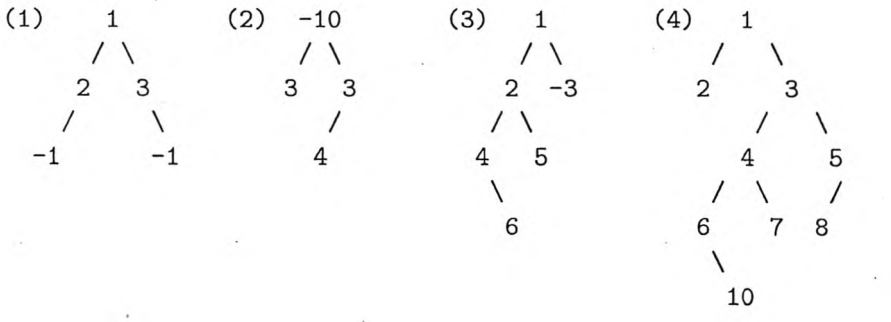
For "!wwkew", there are two longest substrings (i.e., "wke" and "kew") with the length of 3.

Hint As mentioned in the instructions, the time limit for each problem is 1 second. The CPU can execute around 10^9 operations in 1 second. If you use brute force methods to check all the substrings, you may not pass the system.

Source LeetCode 3

Problem F - Binary Tree Maximum Path Sum

A binary tree can be uniquely represented in the following way. For example, given the level order traversals [1 2 3 -1 0 0 -1], [-10 3 3 0 0 4], [1 2 -3 4 5 0 0 0 6], and [1 2 3 0 0 4 5 0 0 0 0 6 7 8 0 0 0 0 0 0 0 0 0 0 10] (0 means no node is here), we can obtain the following trees respectively:



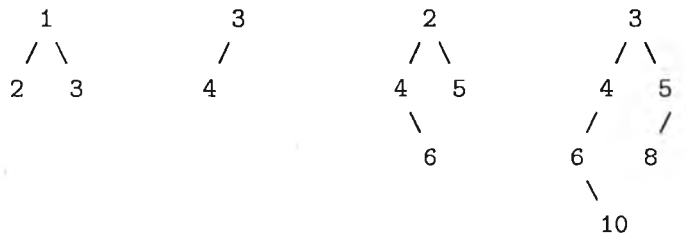
Rules for level order traversal:

(a) Level order traversal of a tree is a breadth first traversal from the root. Breadth first traversal starts at the tree root and explores the neighbor nodes first, before moving to the next level neighbors.

(b) We use '0' to represent that no node is here. For example, the level traversal for the third level in Tree (4) (we regard the root as the 0th level.) is '0 0 0 0 6 7 8 0'.

(c) The traversal is terminated when reaching the last node at the last level (taking the node '10' in Tree (4) as an example).

Given the level order traversal of a tree which has at least one node, you are asked to return the maximum path sum. A path is defined as a sequence of nodes starting from one node to another node along the parent-child connections in the tree. A path must contain at least one node and does not need to go through the root. For example, the maximum sum paths of the former four examples are:



Therefore, the maximum path sums are 6, 7, 17, and 36 respectively.

Input The first line is an integer N ($1 \leq N \leq 10^3$) indicating the number of test cases. Each test case contains two lines. The first line is an integer m ($1 \leq m \leq 10^4$). The second line is a sequence of m integers that describes a binary tree. Each integer is separated by a space and there is no space after the last token. You may assume that the value of each node of the tree is an integer between -10^3 and 10^3 except 0.

Output The maximum path sum of the given binary tree.

Sample Input

```
4
1
2
7
1 2 3 4 0 0 -1
9
1 2 -3 4 5 0 0 0 6
6
-1 -2 -2 0 0 -3
```

Sample Output

```
2
10
17
-1
```

Source LeetCode 124

Problem G - Minimum Cost to Merge Stones

There are n piles of stones arranged in a row. The ith pile has stones[i] stones. A *move* consists of merging exactly k **consecutive** piles into one pile. The cost of this *move* is equal to the total number of stones in these k piles.

You are required to find the minimum cost to merge all piles of stones into **one** pile. If it is impossible, return -1.

Input The first line is an integer N ($1 \leq N \leq 10^3$) indicating the number of test cases.

Each test case contains two lines. The first line contains two integers n ($1 \leq n \leq 10^2$) and k ($1 \leq k \leq n$), which are separated by a space. The second line is the stones array, including n integers. These integers are separated by spaces. There is no extra space at the end of the line.

Output For each test case, output the minimum cost or -1.

Sample Input

```
3
4 2
3 2 4 1
4 3
3 2 4 1
5 3
3 5 1 2 6
```

Sample Output

```
20
-1
25
```

Explanation There are 3 test cases.

For the first test case, We start with [3, 2, 4, 1].

1. We merge [3, 2] for a cost of 5, and the remaining stone piles are [5, 4, 1].
2. We merge [4, 1] for a cost of 5, and the remaining stone piles are [5, 5].
3. We merge [5, 5] for a cost of 10, and the remaining stone pile is [10].

The total cost was 5+5+10=20, and this is the minimum possible.

For the second test case, after any merge operation, there are 2 piles left, and we can't merge anymore. So the task is impossible. We return -1.

For the third test case, we start with [3, 5, 1, 2, 6].

1. We merge [5, 1, 2] for a cost of 8, and the remaining stone piles are [3, 8, 6].
2. We merge [3, 8, 6] for a cost of 17, and the remaining stone pile is [17].

The total cost was 8+17=25, and this is the minimum possible.

Source LeetCode 1000

Dec	Characters	Dec	Characters	Dec	Characters	Dec	Characters
33	!	58	:	83	S	108	l
34	"	59	;	84	T	109	m
35	#	60	<	85	U	110	n
36	$	61	=	86	V	111	o
37	%	62	>	87	W	112	p
38	&	63	?	88	X	113	q
39	'	64	@	89	Y	114	r
40	(	65	A	90	Z	115	s
41	)	66	B	91	[	116	t
42	*	67	C	92	\	117	u
43	+	68	D	93	]	118	v
44	,	69	E	94	^	119	w
45	-	70	F	95	_	120	x
46	.	71	G	96	`	121	y
47	/	72	H	97	a	122	z
48	0	73	I	98	b	123	{
49	1	74	J	99	c	124	|
50	2	75	K	100	d	125	}
51	3	76	L	101	e	126	~
52	4	77	M	102	f		
53	5	78	N	103	g		
54	6	79	O	104	h		
55	7	80	P	105	i		
56	8	81	Q	106	j		
57	9	82	R	107	k		

Figure 1: Characters from ASCLL code 33 (decimal) to 126 (decimal).