Indian Statistical Institute

Semester-I 2021-2022

M.Tech.(CS) - First Year

Lab Test 3 (19 February, 2022)

Subject: Computing Laboratory

Total: 60 marks Duration: 3 hrs.

**SUBMISSION INSTRUCTIONS**

1. Naming convention for your programs: `cs21xx-test3-progy.c` or `cs21xx-test3-progy.py` (assuming `cs21xx` denotes your roll number and `progy` denotes the program number).
2. To submit the solution files (`.c`, `.py` 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 - labtest3`.
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. Let us define a positive number as `SPECIAL` whose sum of digits in left half is equal to the sum of digits in its right half. Notably, if the number of digits is odd then inclusion or exclusion of the middle digit in the left and right halves does not matter. Hence, every number having a single digit is a `SPECIAL` number. For example, the numbers 1 (because 1 = 1), 121 (because 1+2 = 2+1 and also 1 = 1), 1221 (because 1+2 = 2+1), 12321 (because 1+2+3 = 3+2+1 and also 1+2 = 2+1), 123321 (because 1+2+3 = 3+2+1), 1234321 (because 1+2+3+4 = 4+3+2+1 and also 1+2+3 = 3+2+1) are all `SPECIAL`.

Consider the following C program that prints the count of all the `SPECIAL` numbers strictly less than $n$ (read from stdin) with a bug and reports the execution time. Revise the program, by only replacing the portion embraced within /* START OF BLOCK */ and /* END OF BLOCK */, such that the revised version of the program performs without the bug and also better in terms of the time taken. More credit will be given toward more efficient revision.

Note that, the count of `SPECIAL` numbers that are less than 20, 100 and 120 are 10, 18 and 20, respectively.

```
#include<sys/time.h>
#include<stdio.h>
#define GAP(start, end) ((end.tv_usec-start.tv_usec)+(end.tv_sec-start.tv_sec)*1000000)
void main(){
    int i, j, k, r, n, numDIGIT, leftSUM, rightSUM, count;
    struct timeval start_time, end_time;
    scanf("%d", &n);
    gettimeofday(&start_time, NULL);
    /* START OF BLOCK */
    count = 0;
    for(i = 1; i < n; i++){
        j = i;
        numDIGIT = 0;
        leftSUM = 0;
        rightSUM = 0;
        while(j > 0){
            j = j/10;
            numDIGIT++;
        }
        j = i;
        k = numDIGIT;
        while(j > 0){
            r = j%10;
            if(k <= numDIGIT/2 + 1)
                leftSUM = leftSUM + r;
            if(k >= numDIGIT/2 + 1)
                rightSUM = rightSUM + r;
            j = j/10;
            k--;
        }
        if(leftSUM == rightSUM)
            count++;
    }
    /* END OF BLOCK */
    gettimeofday(&end_time, NULL);
    printf("Count of special numbers less than %d: %d", n, count);
    printf("\nTime taken: %ld microseconds", (long int) GAP(start_time, end_time));
}
```

[20]

Q2. A sequence $a_0, a_1, a_2, \ldots, a_{n-1}$ is defined as $\Lambda$-*bitonic* if there exists a $j$, $0 \leq j < n$, such that

$a_0 < a_1 < \cdots < a_j > a_{j+1} > \cdots > a_{n-1}$. Consider an $m \times n$ matrix $A$ consisting of integer entries, such that each row and each column of the matrix forms a $\Lambda$-bitonic sequence. Given a range, write a program to efficiently find the elements of the matrix that fall within the range (inclusive).

**Input Format**

Input will be provided via standard input in the following format. The first line of input consists of a pair of integers, namely the number of rows ($m$) and number of columns ($n$) of the matrix $A$. It follows by the elements of $A$, providing each row in a single line following their order in the matrix. In the final line, the range is given as the minimum bound followed by the maximum bound.

**Output Format**

Output is to be printed on the standard output in the following format. The output will print the elements, which fall within the given range (inclusive), of the input matrix $A$. The elements must be printed in ascending order.

**Sample Input 0**

```
1 10
-8 -3 0 5 11 15 20 10 8 4
5 8
```

**Sample Output 0**

```
5 8
```

**Sample Input 1**

```
2 2
4 2
3 1
1 4
```

**Sample Output 1**

```
1 2 3 4
```

**Sample Input 2**

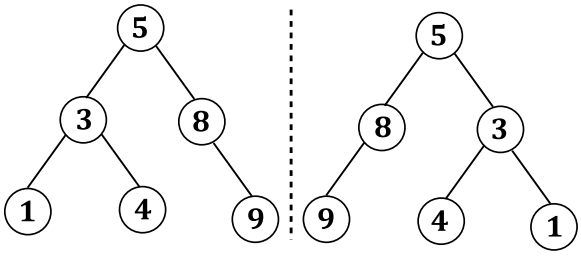
```
4 5
10 19 30 28 26
15 25 35 50 41
12 20 22 38 40
11 19 22 38 45
11 20
```

**Sample Output 2**

```
11 12 15 19 20
```

Q3. Let us define a binary tree as the *mirror twin* of a binary search tree (BST) if both of them comprises the same set of data items and are structurally mirror (while the mirror is placed vertically) to each other. Given the preorder traversal of a BST and the inorder and postorder traversals of a simple binary tree as user inputs, write a program to determine whether the binary tree is a mirror twin of the BST or not.

As for example, the binary tree shown below (in the rightside) is a mirror twin of the BST given below (in the leftside).




**Input Format**

Input will be provided via standard input in the following format. The first line of input consists of two integers, namely the number of data items in both the input trees. It follows by three more input lines. These lines consist of sets of integers corresponding to the data items obtained from the preorder traversal on the BST, followed by the inorder and postorder traversals on the binary tree, respectively.

**Output Format**

Output is to be printed on the standard output in the following format. The output simply prints 'MIRROR TWIN' or 'NOT MIRROR TWIN' corresponding to whether the input binary tree is a mirror twin to the BST or not.

**Sample Input 0**

```
6 6
5 3 1 4 8 9
1 3 4 5 8 9
5 3 1 4 8 9
```

**Sample Output 0**

```
NOT MIRROR TWIN
```

**Sample Input 1**

```
6 6
5 3 1 4 8 9
9 8 5 4 3 1
9 8 4 1 3 5
```

**Sample Output 1**

```
MIRROR TWIN
```

**Sample Input 2**

```
5 5
8 6 3 2 4
8 6 4 3 2
8 6 3 4 2
```

**Sample Output 2**

```
NOT MIRROR TWIN
```