**PROJECT REPORT**

|  |  |
| --- | --- |
| **Course Code and Name:**  CSE 325  Operating System | |
| **Project name:**  Multithreaded Web Server Implementation | |
| **Name of Student:**  Adri Saha  (2019-1-60-024) | **Course Instructor information:**  Tanni Mitra  Senior Lecturer  Department of Computer Science & Engineering |
| **Date of Submission**  2 June 2021 | |

**Project Statement**: Implementation of Multithreaded Web Server which can connect with multiple clients using threads. Which means it is capable to take multi-client request and work more efficiently.

**Abstract:**

A multithreaded server is any server that has more than one thread.Multithreaded servers have several transports since each transfer requires its own thread. In this project, we had to build a multithreaded web server.

The main feature of our project is to build a problem solver server client community. Where the multiple clients can discuss between all the client, and they also can discuss it with the server by individually. And the multiple clients under a server can communicate with the server at a time. They will choose their problem and send it to the server and then server’s system takes some input of that problem and send the output of that problem. If any doubts about that problem, then clients can also discuss with the server or with another client about that problem. We also arranged a system to send the editorial file to the clients so that clients can visualize the problems more easily.

**Equipment and Components:**

1. Java, NetBeans software
2. Notepad++ software
3. Operating system ubuntu, windows

**Objective:**

* It is use for minimization and more efficient use of computing resources.
* Requests from one thread do not block requests from other threads, which improves application responsiveness.
* Multithreading takes fewer resources than operating multiple processes at once.

**Introduction:**

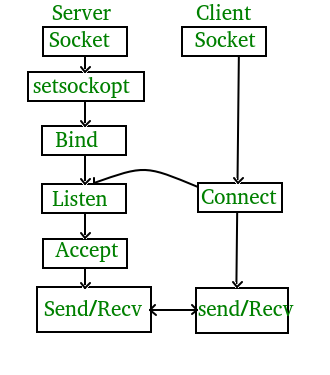
A thread is a flow of execution through the process code that has its own program counter to keep track of which instruction to execute next, system registers to maintain its current working variables, and a stack to keep track of the execution history. A lightweight process is sometimes known as a thread. The ability of a central processing unit (CPU) (or a single core in a multi-core processor) to offer many threads of execution simultaneously, as enabled by the operating system, is known as multithreading.

To start discussion about our project first want to discuss about our concept of socket programming.

**Socket programming:**

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.

The diagram of the server client model is given below:



**Stages for the sever:**

To connect to another machine, we need a socket connection. A socket connection means the two machines have information about each other’s network location (IP Address) and TCP port. The java.net.Socket class represents a Socket. To open a socket:

Socket socket = new Socket(“127.0.0.1”, 5000)

First argument – **IP address of Server**. (127.0.0.1 is the IP address of localhost, where code will run on single stand-alone machine).

Second argument – **TCP Port**. (Just a number representing which application to run on a server. For example, HTTP runs on port 80. Port number can be from 0 to 65535)

**Communication:**

To communicate over a socket connection, streams are used to both input and output the data.

**Closing the connections**:

The socket connection is closed explicitly once the message to server is sent. In the program, Client keeps reading input from user and sends to the server until “Over” is typed.

**Important points**:

Server application makes a ServerSocket on a specific port which is 5000. This starts our Server listening for client requests coming in for port 5000.

Then Server makes a new Socket to communicate with the client.

socket = server.accept()

The accept() method blocks(just sits there) until a client connects to the server.

Then we take input from the socket using getInputStream() method. Our Server keeps receiving messages until the Client sends “Over”.

After we’re done we close the connection by closing the socket and the input stream.

To run the Client and Server application on your machine, compile both. Then first run the server application and then run the Client application.

**To run on terminal or command prompt:**

Open two windows one for Server and another for Client

1. First run the Server application as,

$ java Server

Server started

Waiting for a client …

2. Then run the Client application on another terminal as,

$ java Client

It will show – Connected and the server accepts the client and shows,

Client accepted

3. Then you can start typing messages in the Client window. Here is a sample input to the Client

Hello

I made my first socket connection

Over

Which the Server simultaneously receives and shows,

Hello

I made my first socket connection

Over

Closing connection

Notice that sending “Over” closes the connection between the Client and the Server just like said before.

If you’re using Eclipse or likes of such-

Compile both of them on two different terminals or tabs

Run the Server program first

Then run the Client program

Type messages in the Client Window which will be received and showed by the Server Window simultaneously.

Type Over to end.

The additional problem’s we take from the codeforces, and the name and the link of that problem’s are given below:

1: D.Corrupted Array(https://codeforces.com/problemset/problem/1512/D)

2: D.Same Difference(https://codeforces.com/contest/1520/problem/D)

3: D.Position's in the array(past time’s we solve that but cannot find link)

4: D. Maximum Sum of Products(https://codeforces.com/problemset/problem/1519/D)

5: C.Double Ended String(https://codeforces.com/problemset/problem/1506/C)

6: C.Similar Pairs(https://codeforces.com/problemset/problem/1360/C)

7: C.Not Adjacent Matrix(https://codeforces.com/problemset/problem/1520/C)

8: C.Board Moves(https://codeforces.com/problemset/problem/1353/C)

9: B. Sifid and Strange Subsequences(https://codeforces.com/problemset/problem/1529/B)

10: B1. Palindrome Game (easy version)( https://codeforces.com/problemset/problem/1527/B1)

11: B. Nastia and a Good Array(https://codeforces.com/problemset/problem/1521/B)

12: B.Prime Square(https://codeforces.com/problemset/problem/1436/B)

13: A. Eshag Loves Big Arrays(https://codeforces.com/problemset/problem/1529/A)

14: A. Nastia and Nearly Good Numbers(https://codeforces.com/problemset/problem/1521/A)

15: A. Red and Blue Beans(https://codeforces.com/problemset/problem/1519/A)

16: A.GCD Sum(https://codeforces.com/problemset/problem/1498/A)

**Multi thread**:

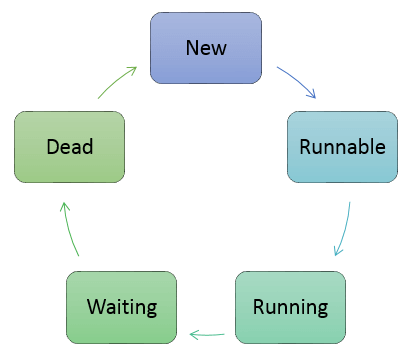
MULTITHREADING in Java is a process of executing two or more threads simultaneously to maximum utilization of CPU. Multithreaded applications execute two or more threads run concurrently. Hence, it is also known as Concurrency in Java. Each thread runs parallel to each other. Multiple threads don't allocate separate memory area, hence they save memory. Also, context switching between threads takes less time.

**Advantages of multithread**:

The users are not blocked because threads are independent, and we can perform multiple operations at times

As such the threads are independent, the other threads won't get affected if one thread meets an exception.

Thread life cycle works as like as like this is:



There are various stages of life cycle of thread as shown in above diagram:

New

Runnable

Running

Waiting

Dead

New: In this phase, the thread is created using class "Thread class".It remains in this state till the program starts the thread. It is also known as born thread.

Runnable: In this page, the instance of the thread is invoked with a start method. The thread control is given to scheduler to finish the execution. It depends on the scheduler, whether to run the thread.

Running: When the thread starts executing, then the state is changed to "running" state. The scheduler selects one thread from the thread pool, and it starts executing in the application.

Waiting: This is the state when a thread has to wait. As there multiple threads are running in the application, there is a need for synchronization between threads. Hence, one thread has to wait, till the other thread gets executed. Therefore, this state is referred as waiting state.

Dead: This is the state when the thread is terminated. The thread is in running state and as soon as it completed processing it is in "dead state"

Some commonly use method in the thread are given below:

start() This method starts the execution of the thread and JVM calls the run() method on the thread.

Sleep(int milliseconds) This method makes the thread sleep hence the thread's execution will pause for milliseconds provided and after that, again the thread starts executing. This help in synchronization of the threads.

getName() It returns the name of the thread.

setPriority(int newpriority) It changes the priority of the thread.

yield () It causes current thread on halt and other threads to execute.

**Code description’s**:

Sever.java: Mainly this class will build the communication between the client’s. with the help of this class multiple client can communicate between themselves’s.

Here we use this kind’s of method to build our system’s

Socket socket;//for our socket connection

DataInputStream din;//get the input of the other system

DataOutputStream dout;//give the output

ServerSocket serverSocket;//this is our key socket

List<ServerConnection> list=new ArrayList<>();//this is the list of the connection’s of the client

Here we have a empty constructor form that we will do that task’s:

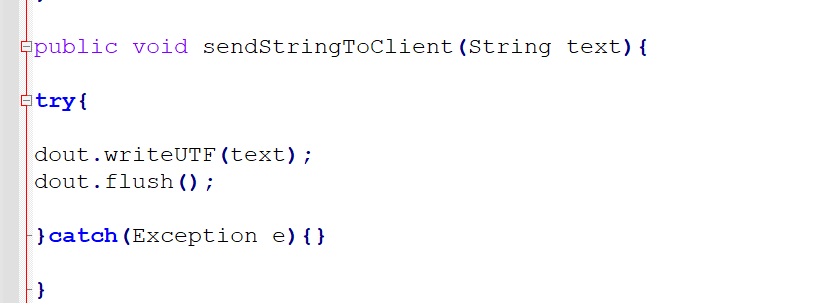
1: we make a server socket connection

2: accept the socket

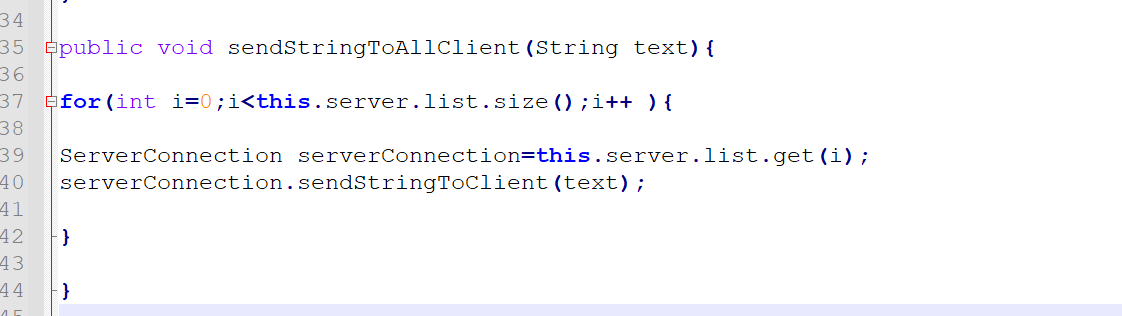
3: start the server connection

4: and add that connection’s in the list

**ServerConnections**: this is actually a thread from that we complete the task’s of the connect between the server and socket.

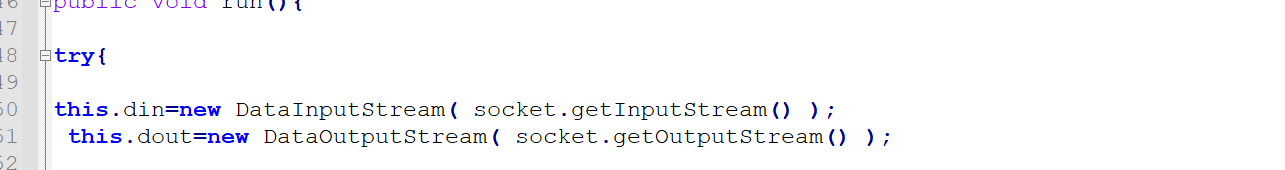


With this method we write message in the client pad

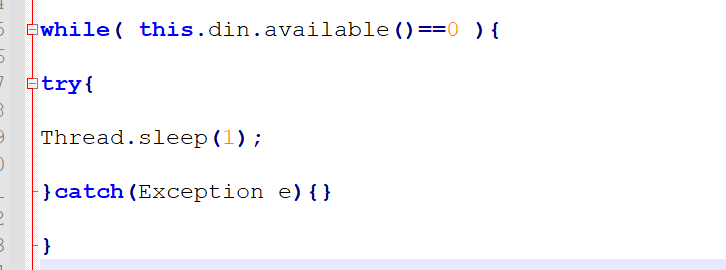


With this send our message to the all client’s socket

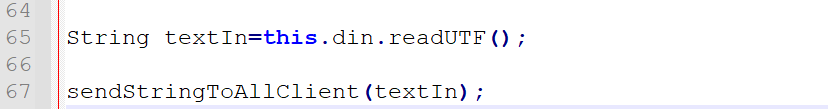
And we have also a run() override methods. Tha task’s of the method is given below:



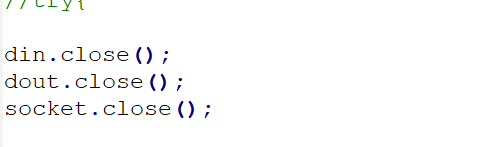
Make the connection ableavail of the read and write operation’s



Read the input of the client

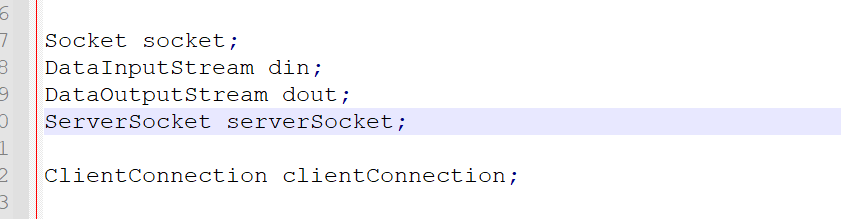


read that utf and send it to the client



Finally close the connection’s

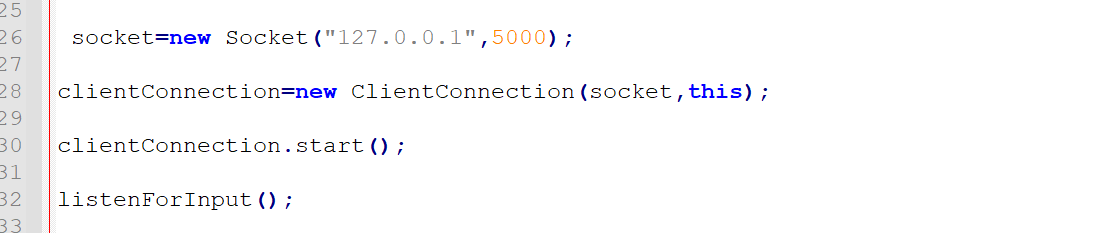
**Client**: This thread will actually works under the sever class or thread



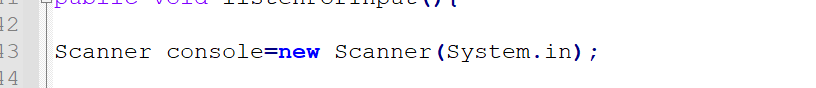
This is the variable’s



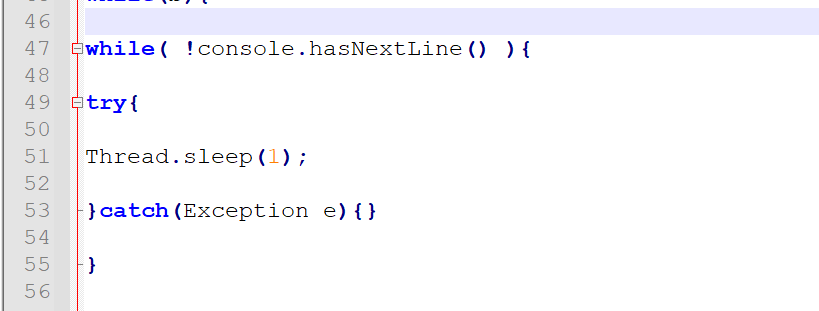
This is an empty constructor



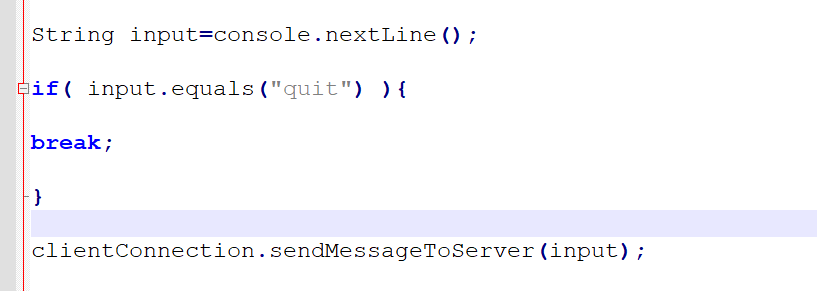
Make a socket connection’s and call the finction for listen the input



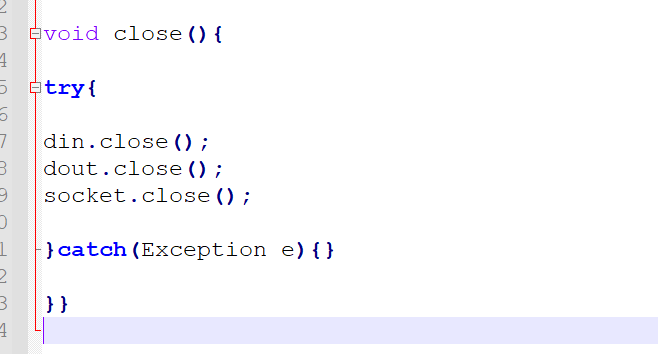
Scanner call to scaned the text’s



While get the input

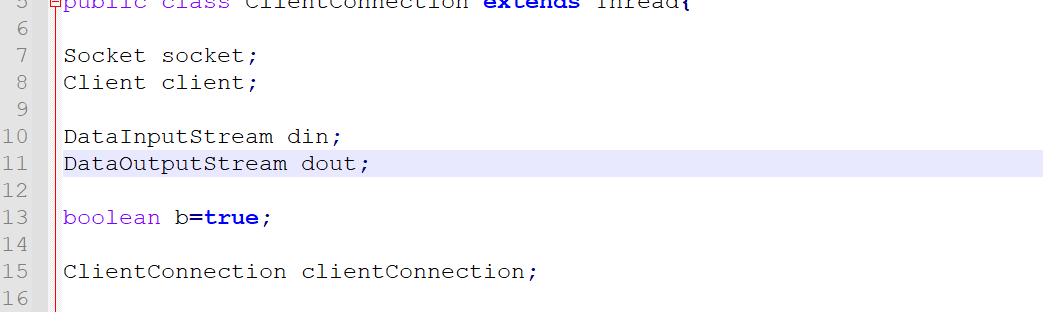


Check anyone want to close or not and send the message

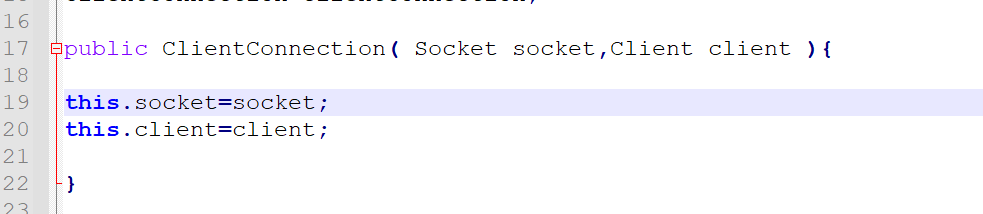


Finally close all the variable’s with the help of that function’s

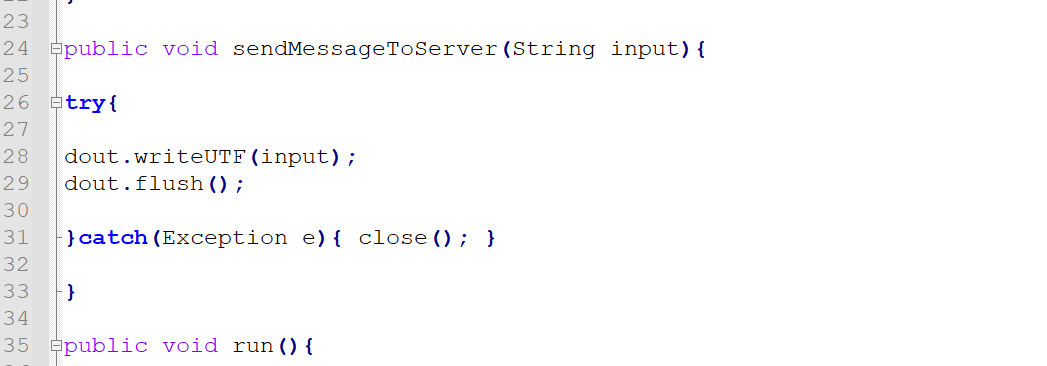
**ClientConnection:**

****

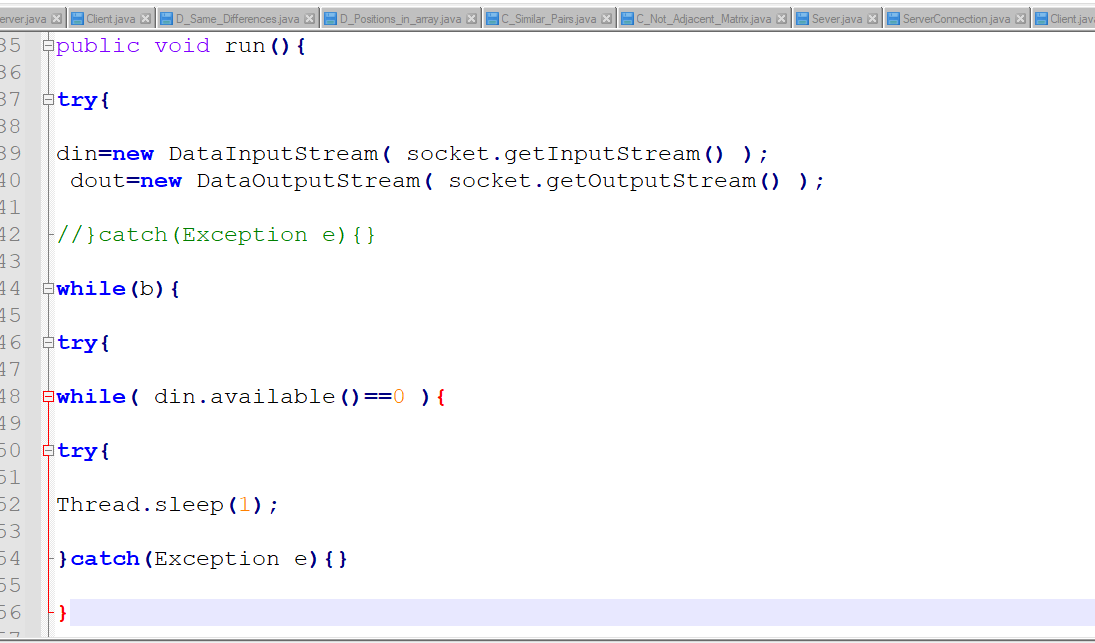
**This is the varibale’s of our method**

****

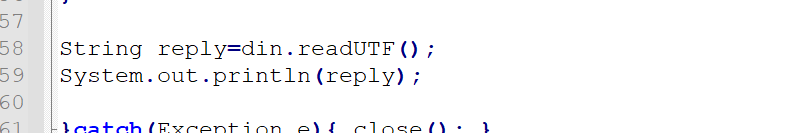
**Set our class variable’s**

****

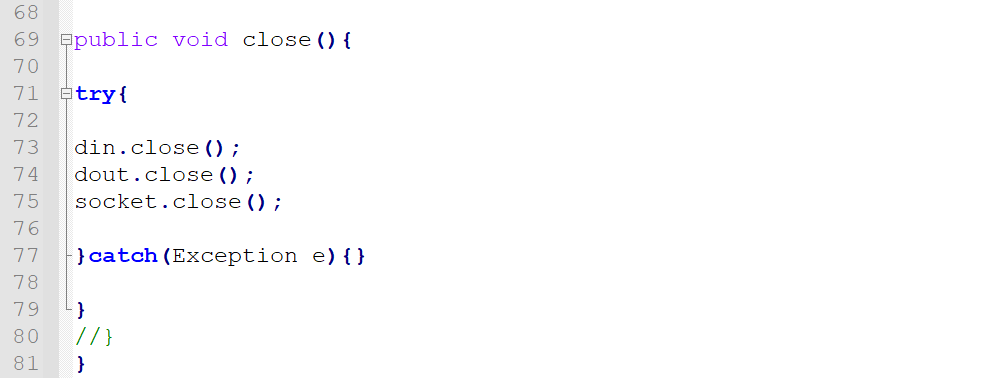
With this we send the message to the server



This is our run the overwrite method. And searching till the input found



And print it



After every operation’s we close the system’s with the help of this function’s

**Server and Client1 class**

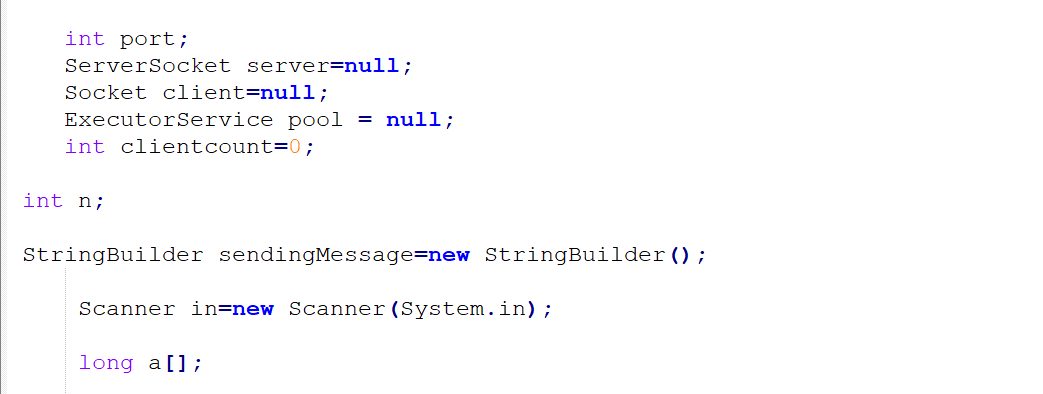
From these two classes we just make a private connection of the server and client class. So that they can discuss about that problem in privately.

The discussion of that problem’s or making algorithm is complete. Now it’s time to solve those problems. To do that we build another folder tcp and there we do the task of the solve the problem or execute that idea:

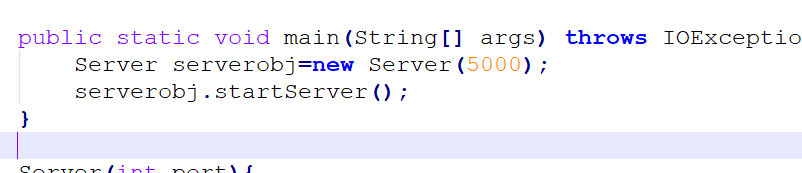
**Server:**



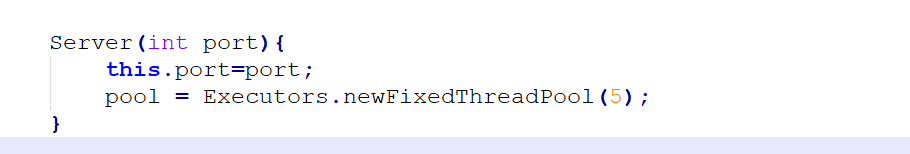
This is our header file’s.



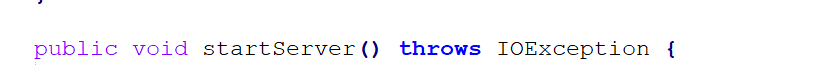
This is our parameter’s



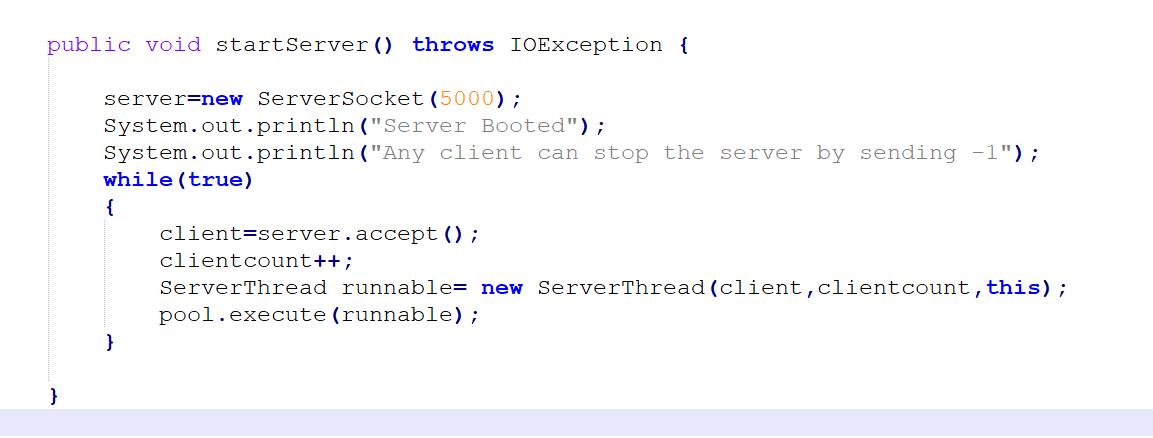
Process to create socket and start the process.



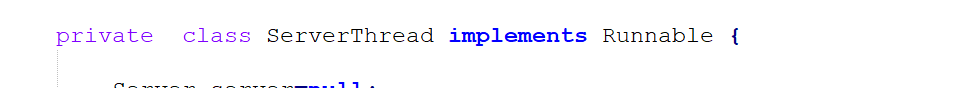
Our connection is build by this constructor and make smothy our connection with executorpoolservice and we push a new thread by this method.



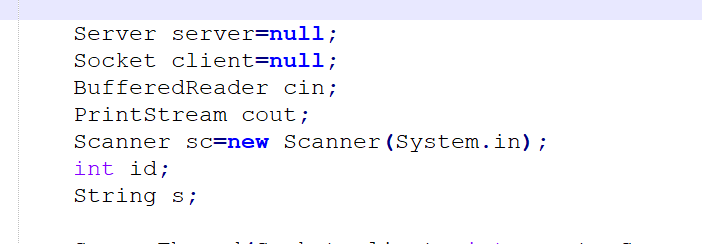
This method usually start the process’s or start the task’s of the server



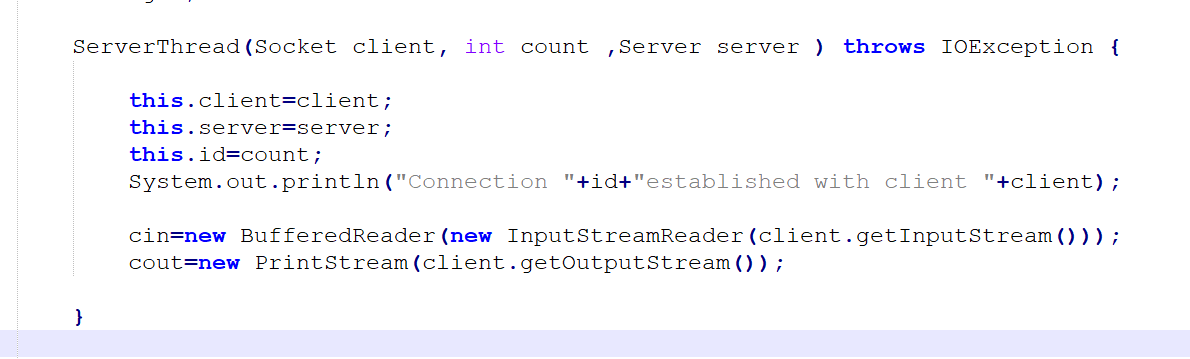
By this method we receive socket’s and while the client send the request we accept that and pool it by the executor pool service method.



This inner class mainly implements the runnable override method

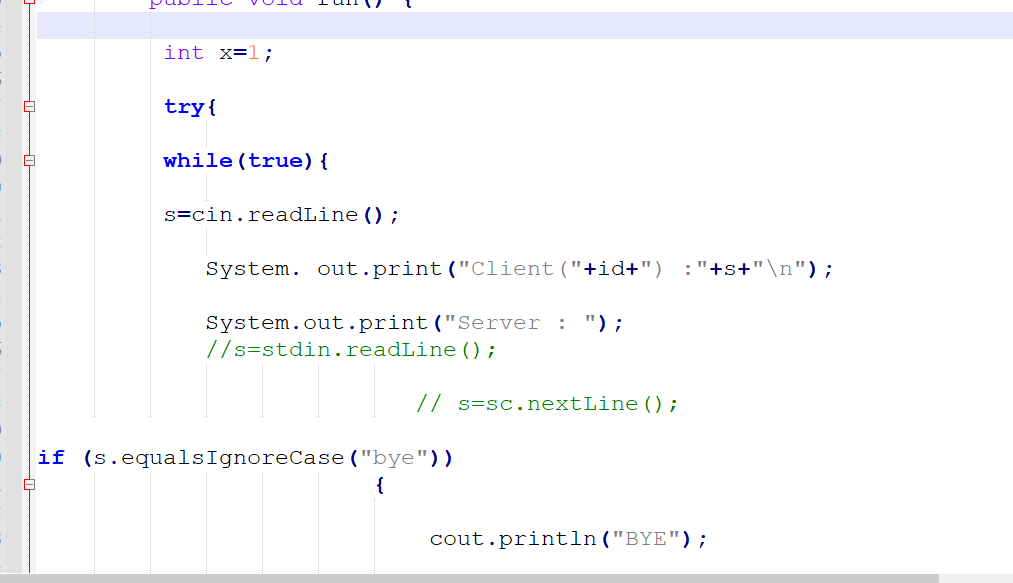


This is mainly our variable’s of this class



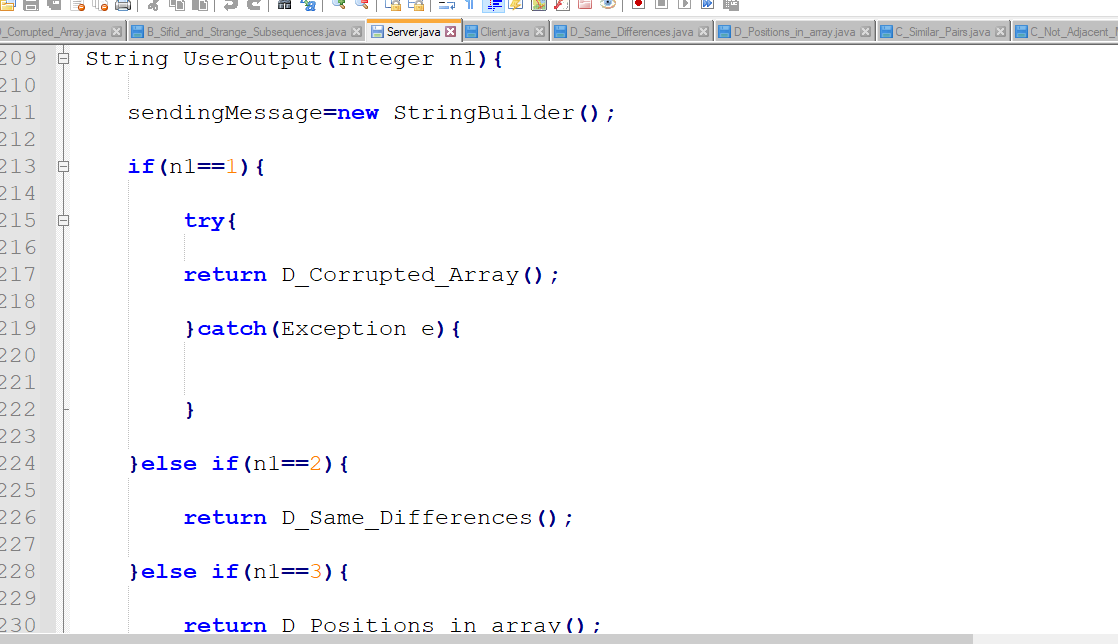
With this constructor we set or prepare our basic steps of all states or variable’s

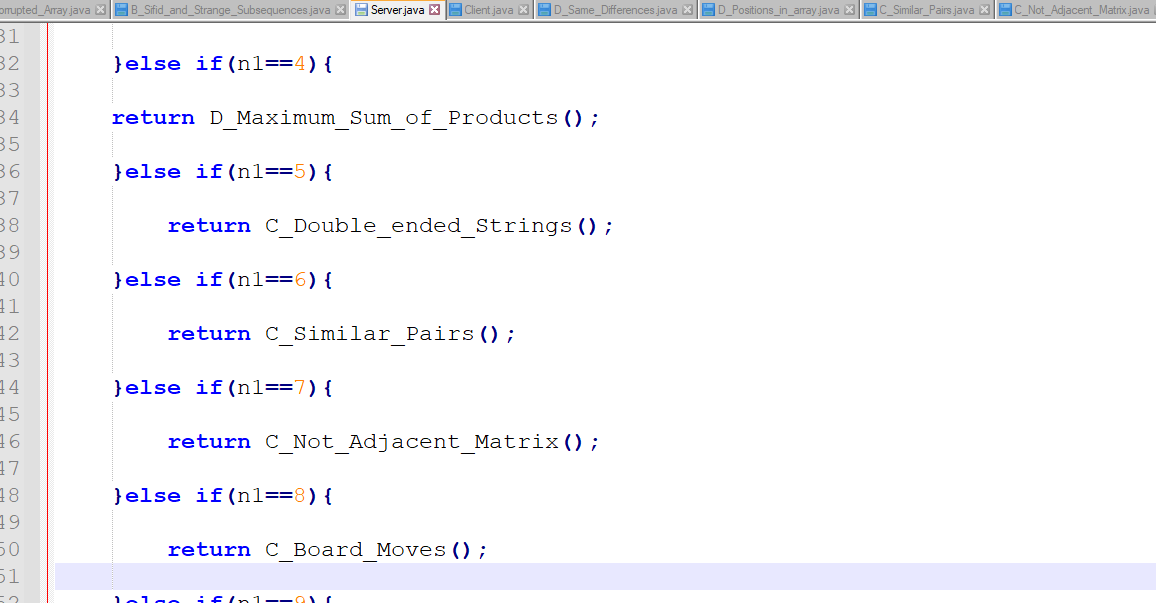
In the run overright method we do this task’s

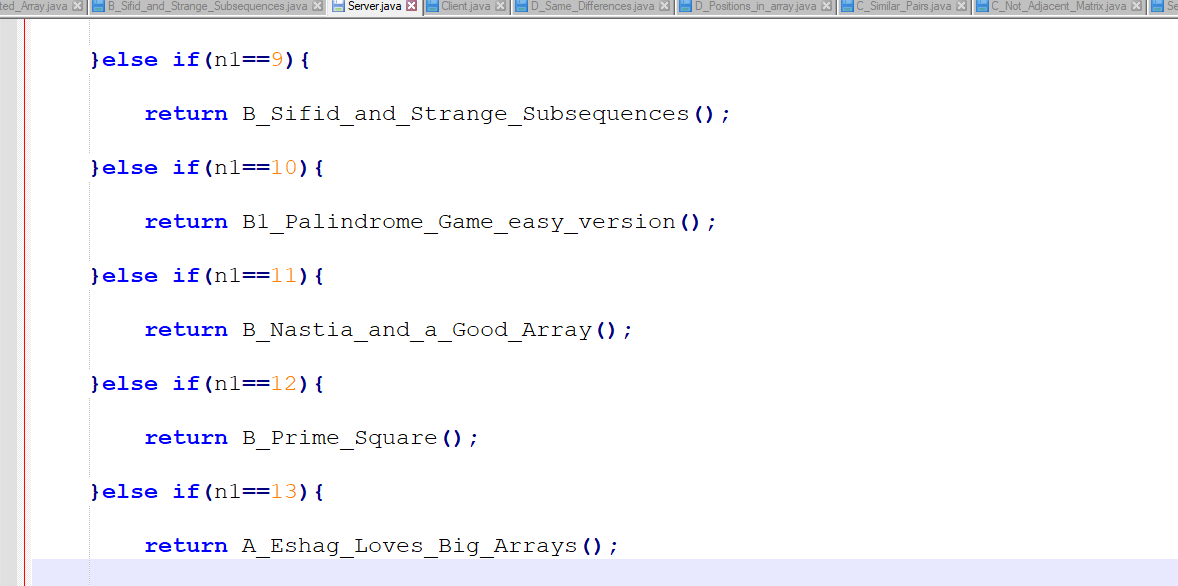


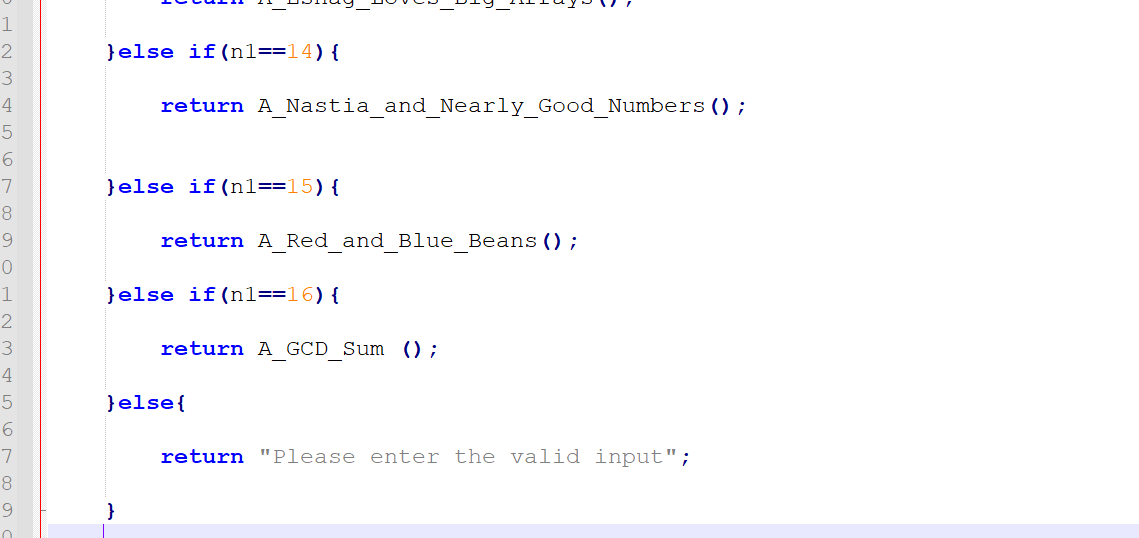
Every time read the input from the opposite, and every state it check that who want to close his/her connection’s and close that connection’s

And then it start the process of the solving, and after that it will close all the system’s





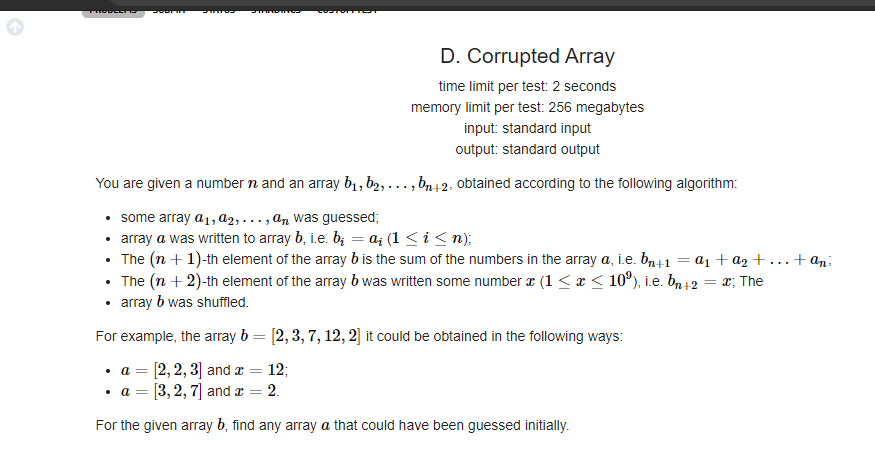




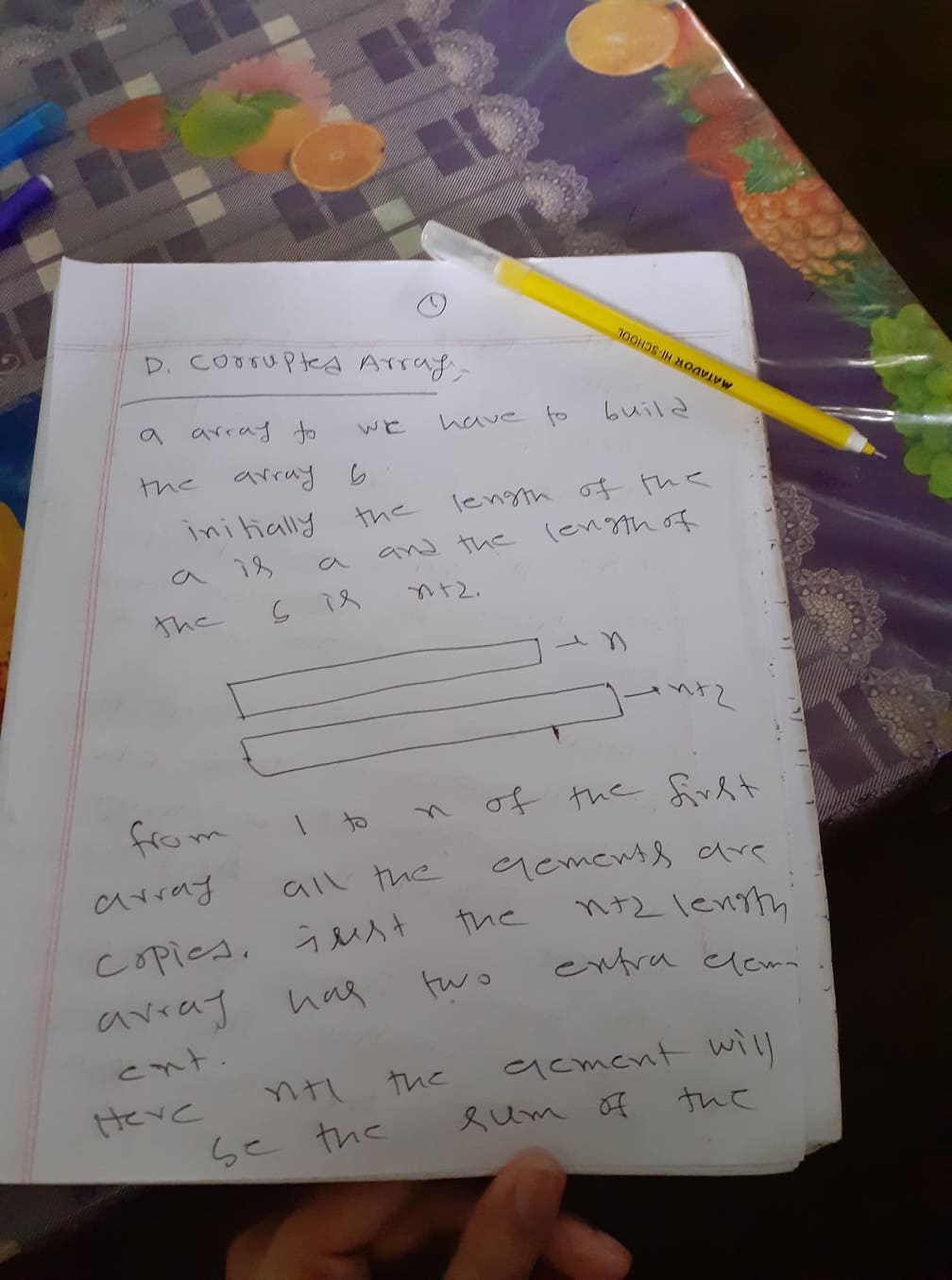
From this line’s of the code we check that what problem user want to solve.

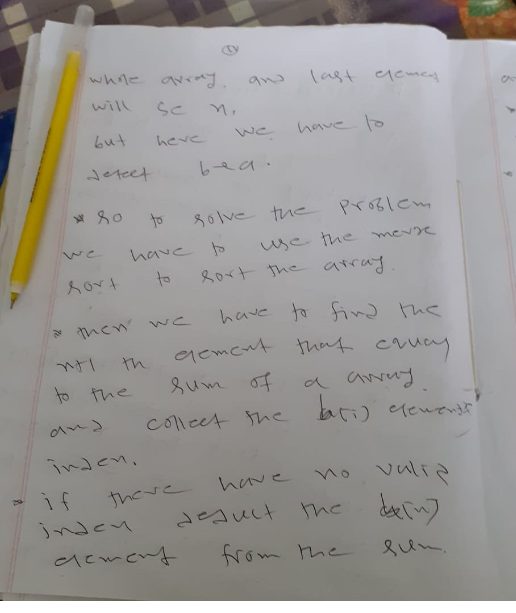
To solve the problem D. Corrupted Array(<https://codeforces.com/problemset/problem/1512/D>)

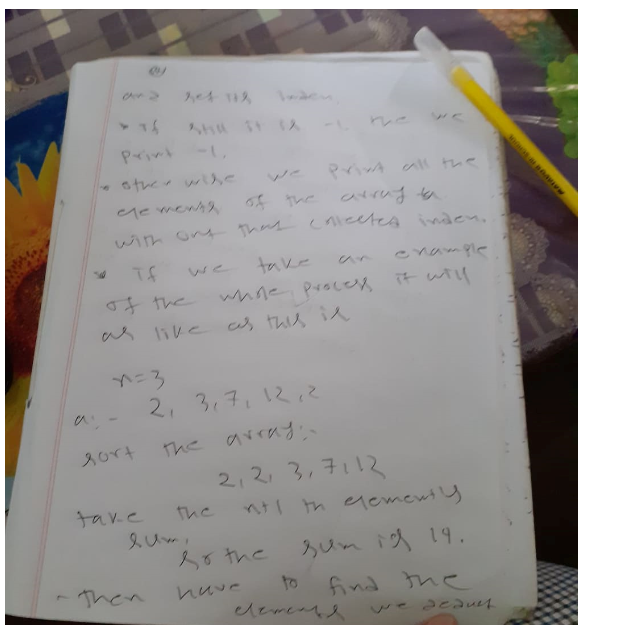
The problem statement is given below:

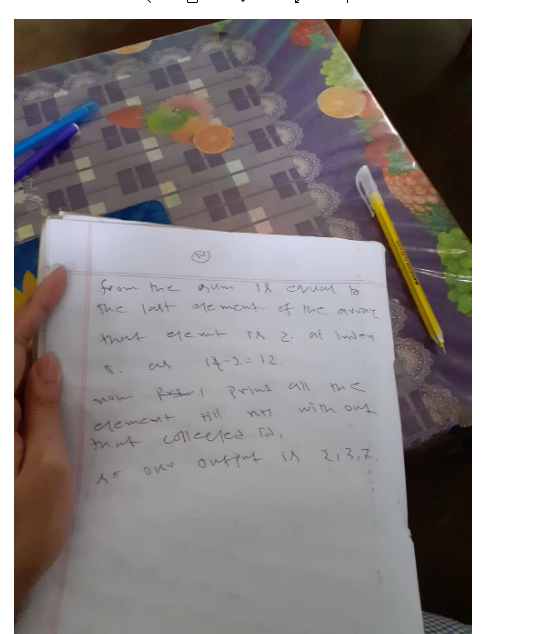


Our process is given below:



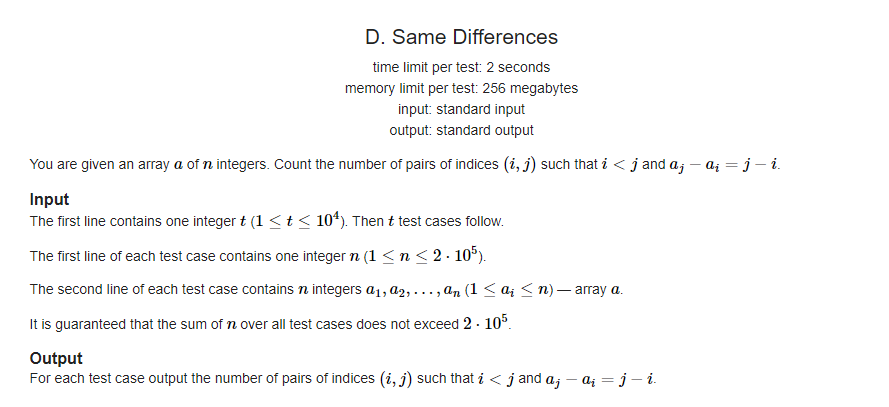




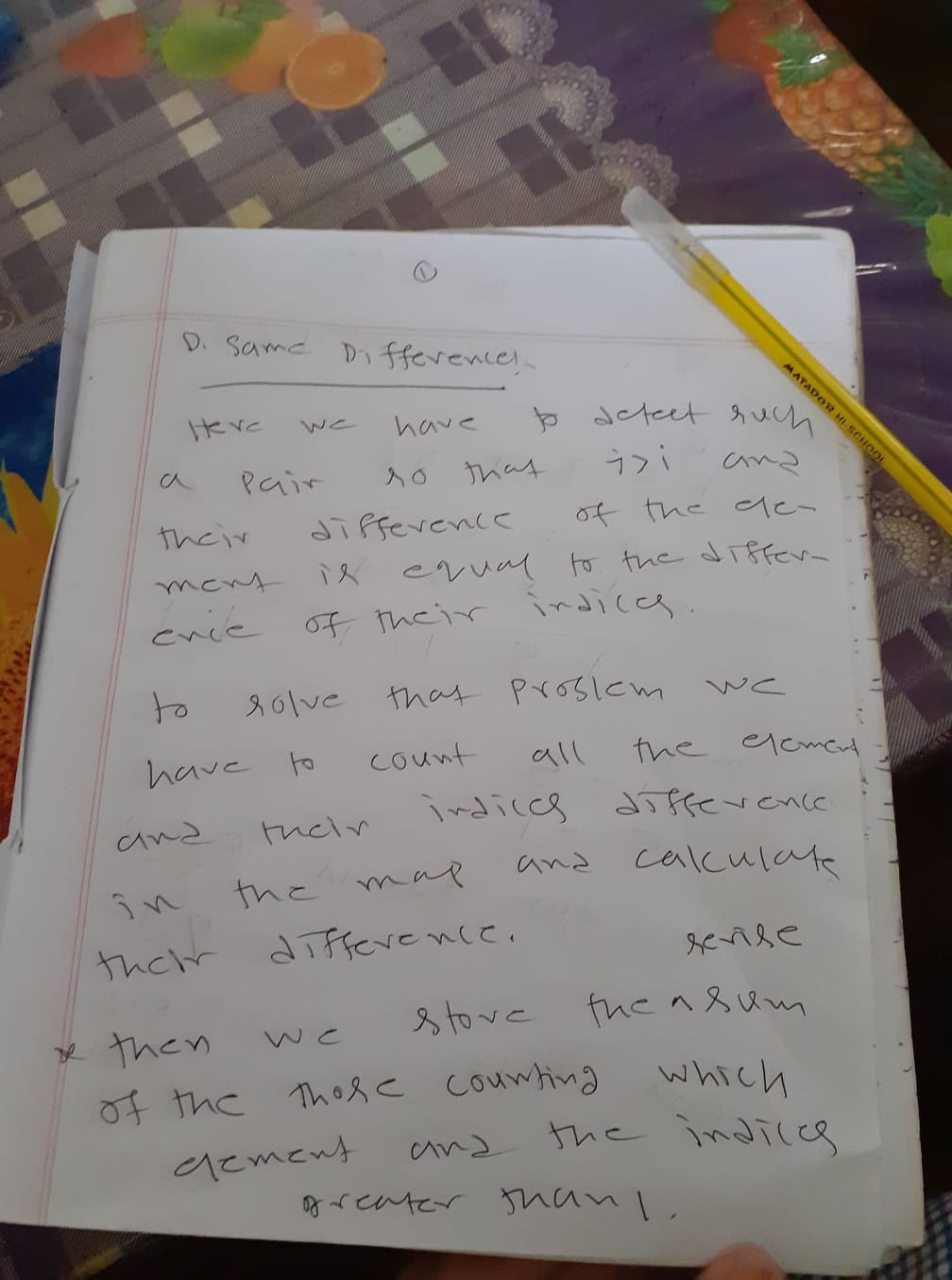


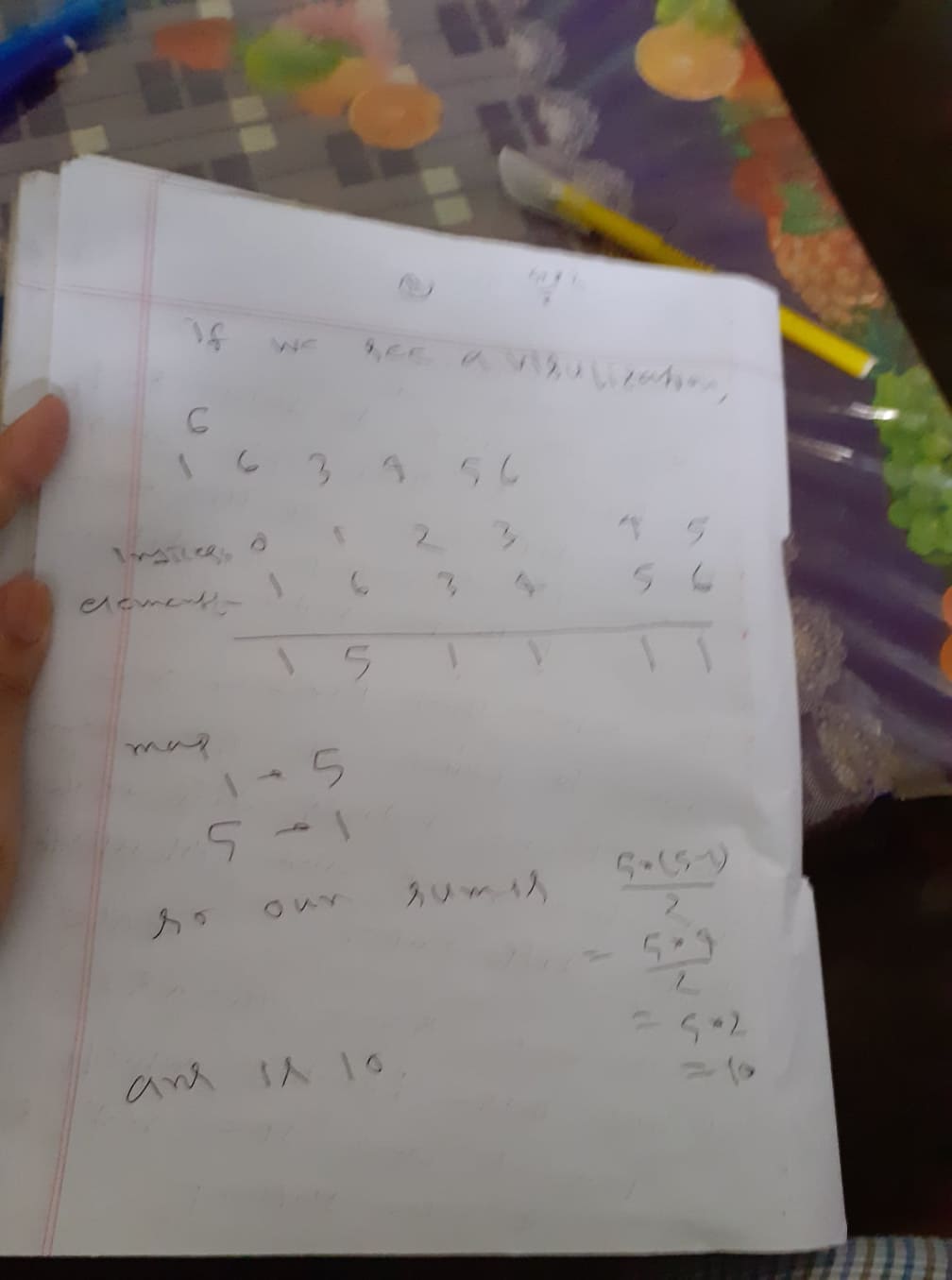
To solve the problem D. Same difference(https://codeforces.com/contest/1520/problem/D):

This is our problem statement:



Our solving way is:



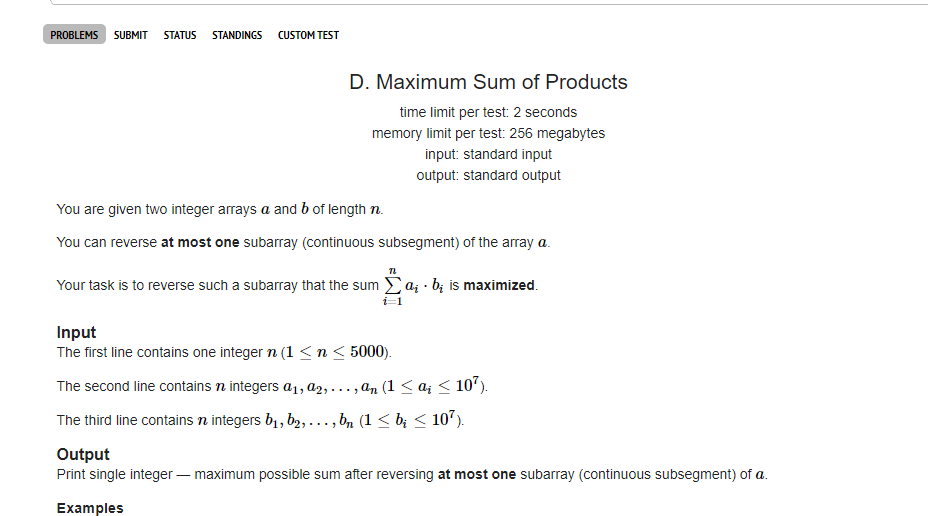


**D. Position’s in the array:**

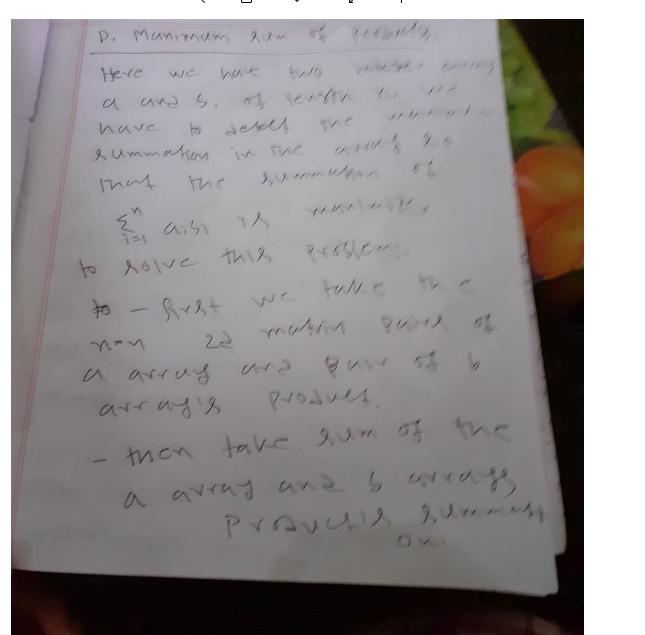
This is not to much just print the element and their position;s in the array

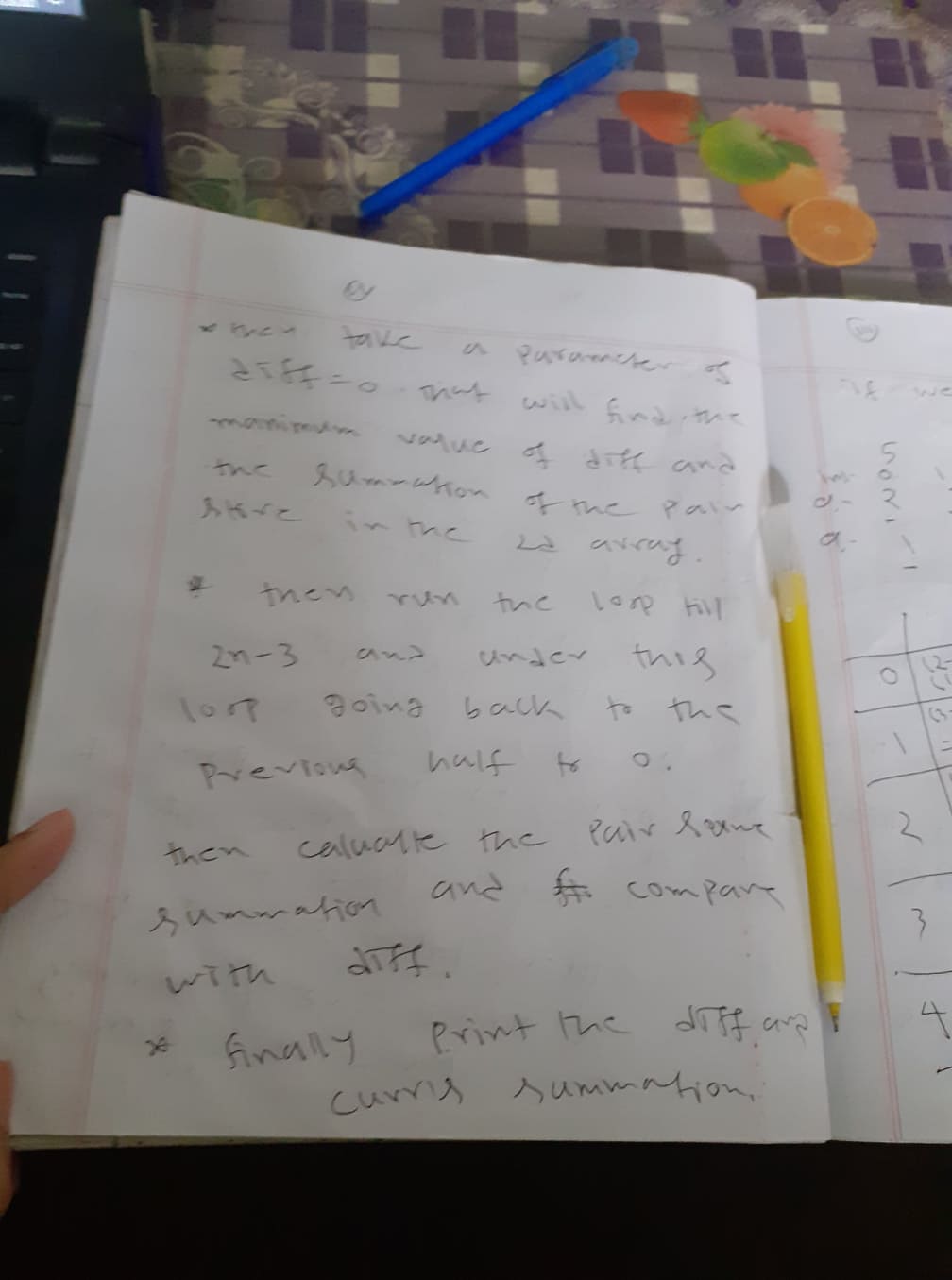
D. Maximum Sum of Products(https://codeforces.com/problemset/problem/1519/D)

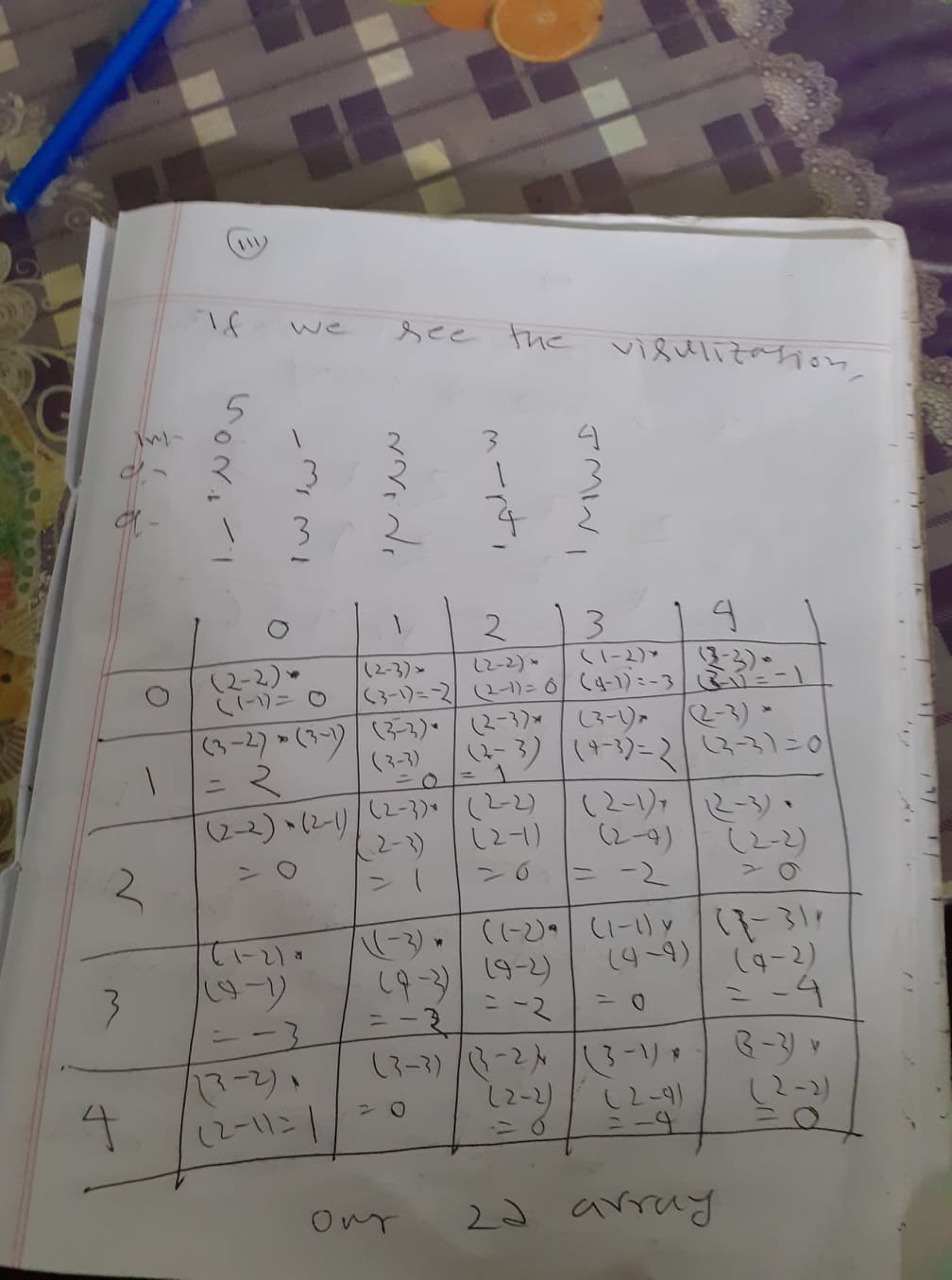
The problem statement is given below:

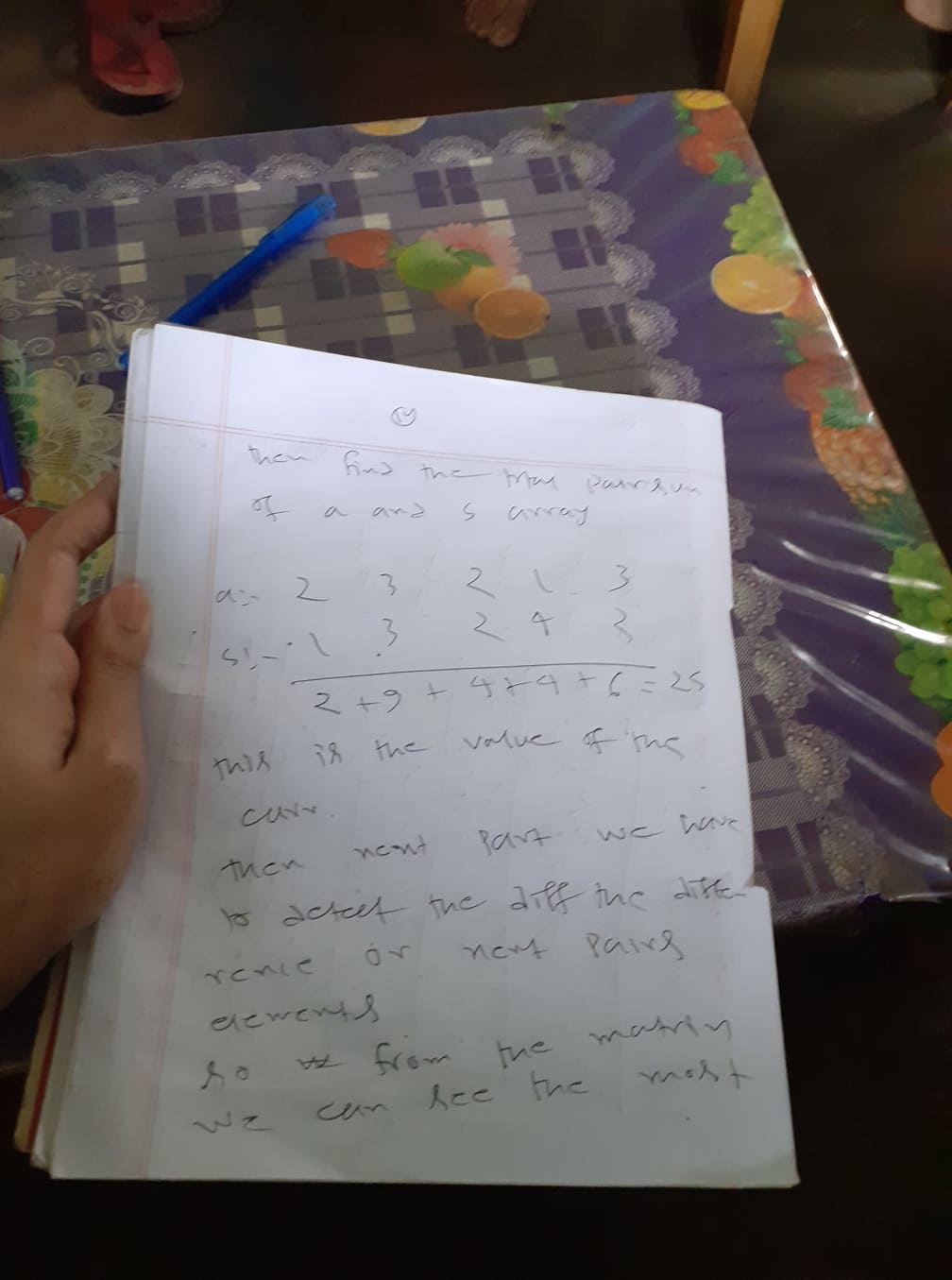


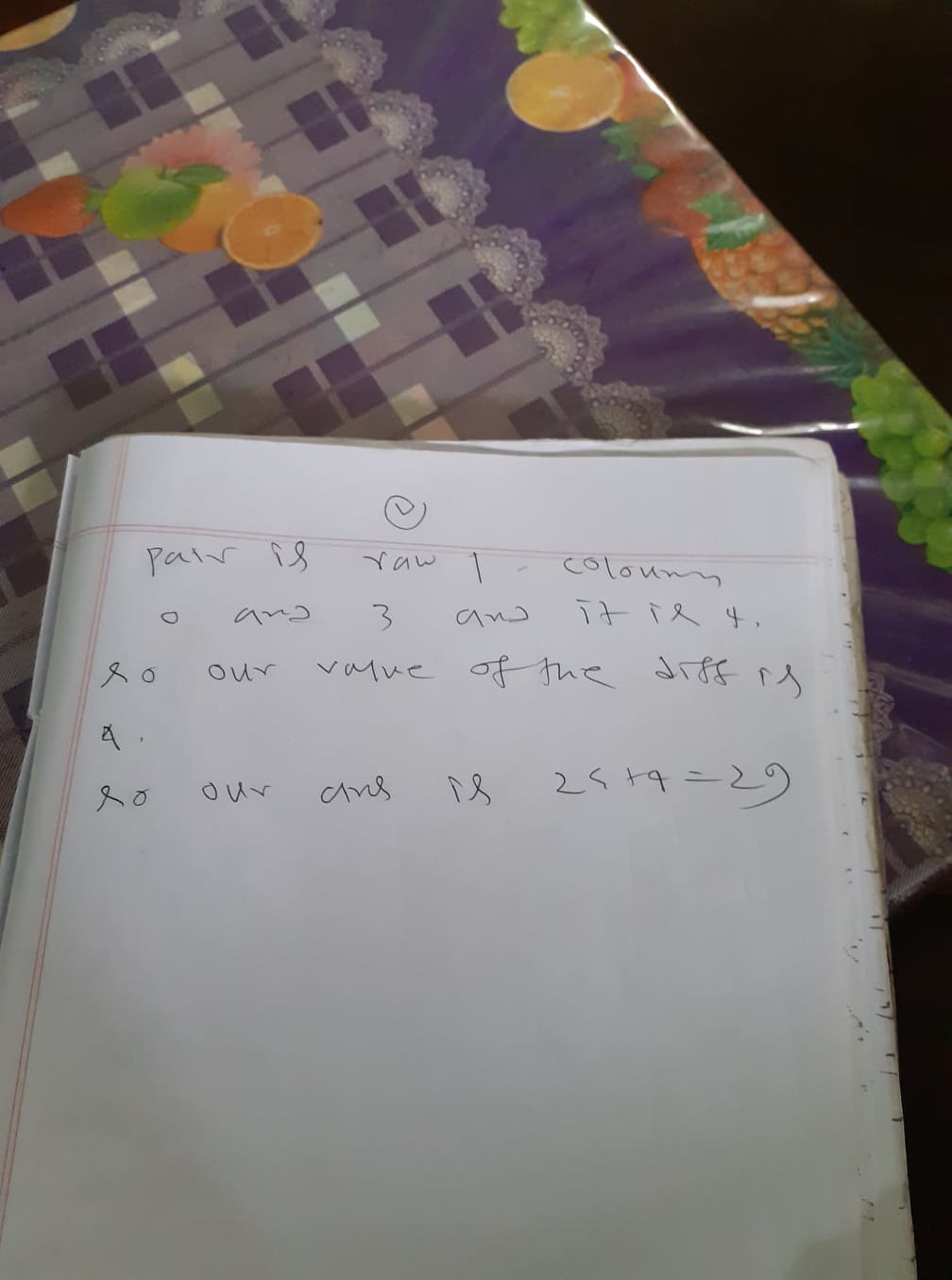
Our solution process is given below:





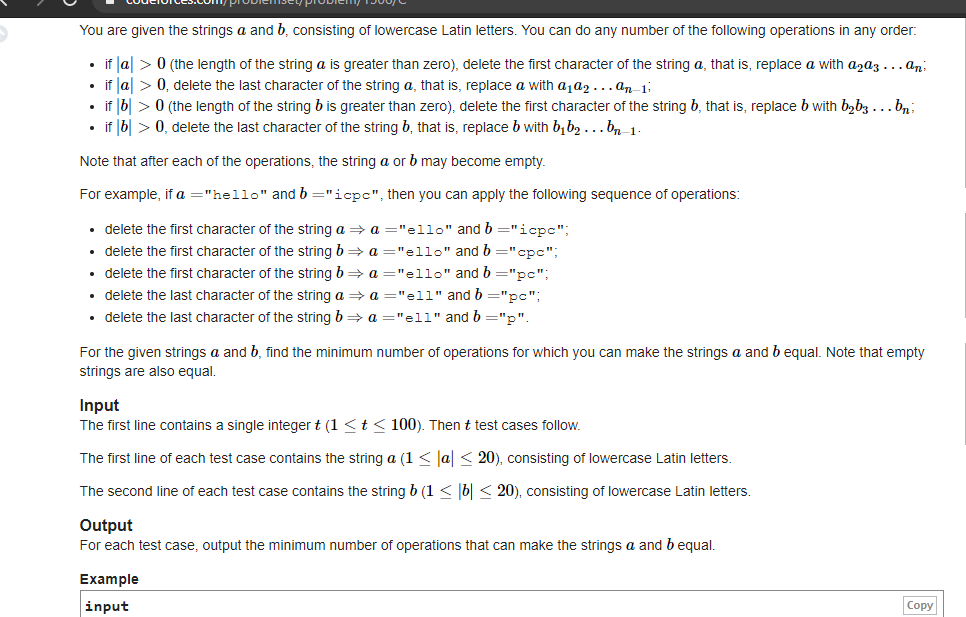




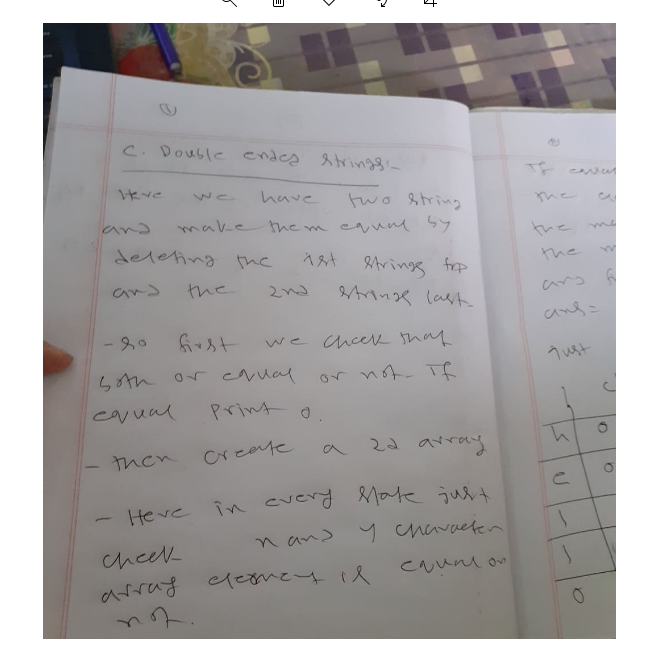


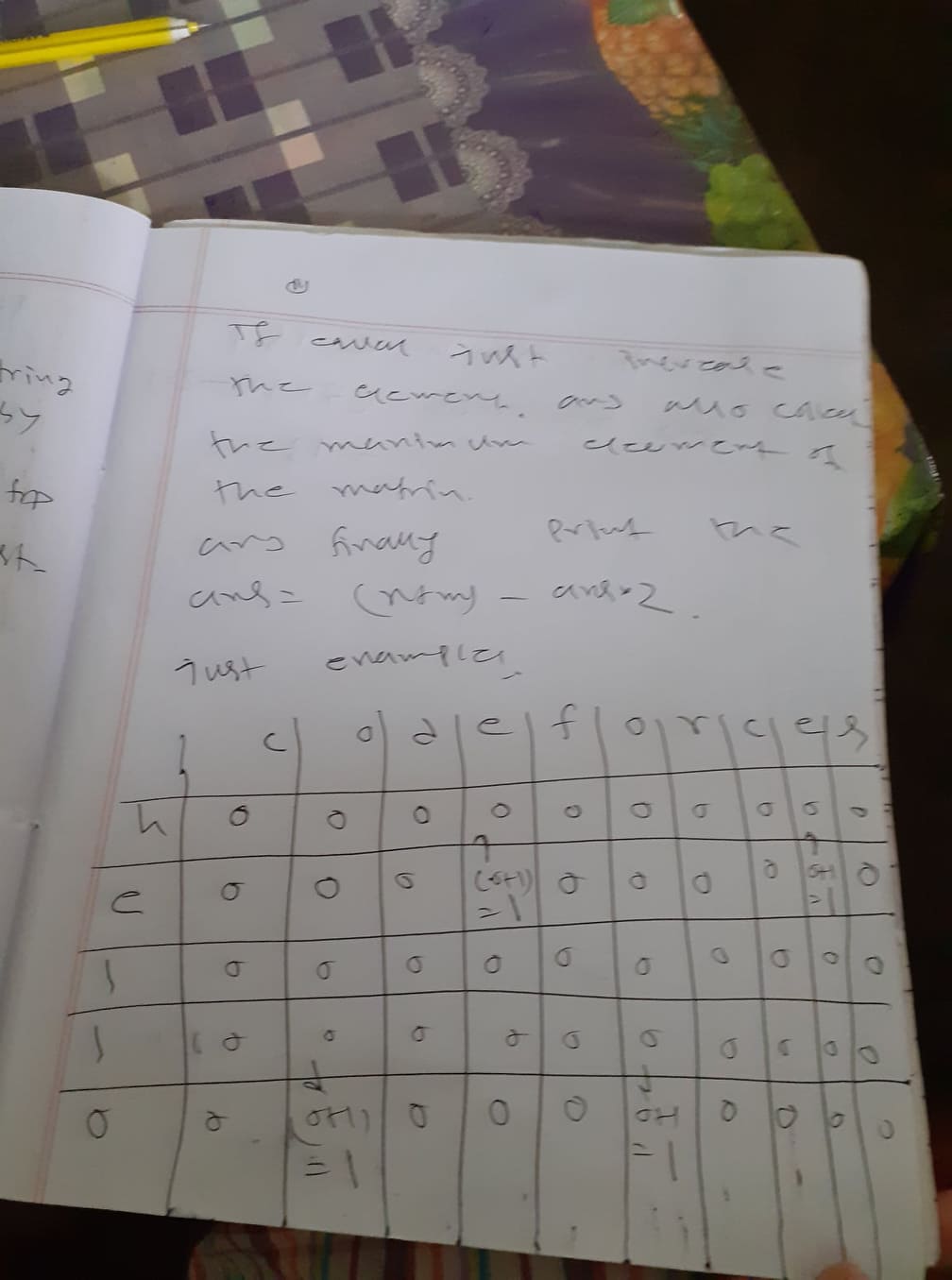
C. Double-ended Strings(<https://codeforces.com/problemset/problem/1506/C>)

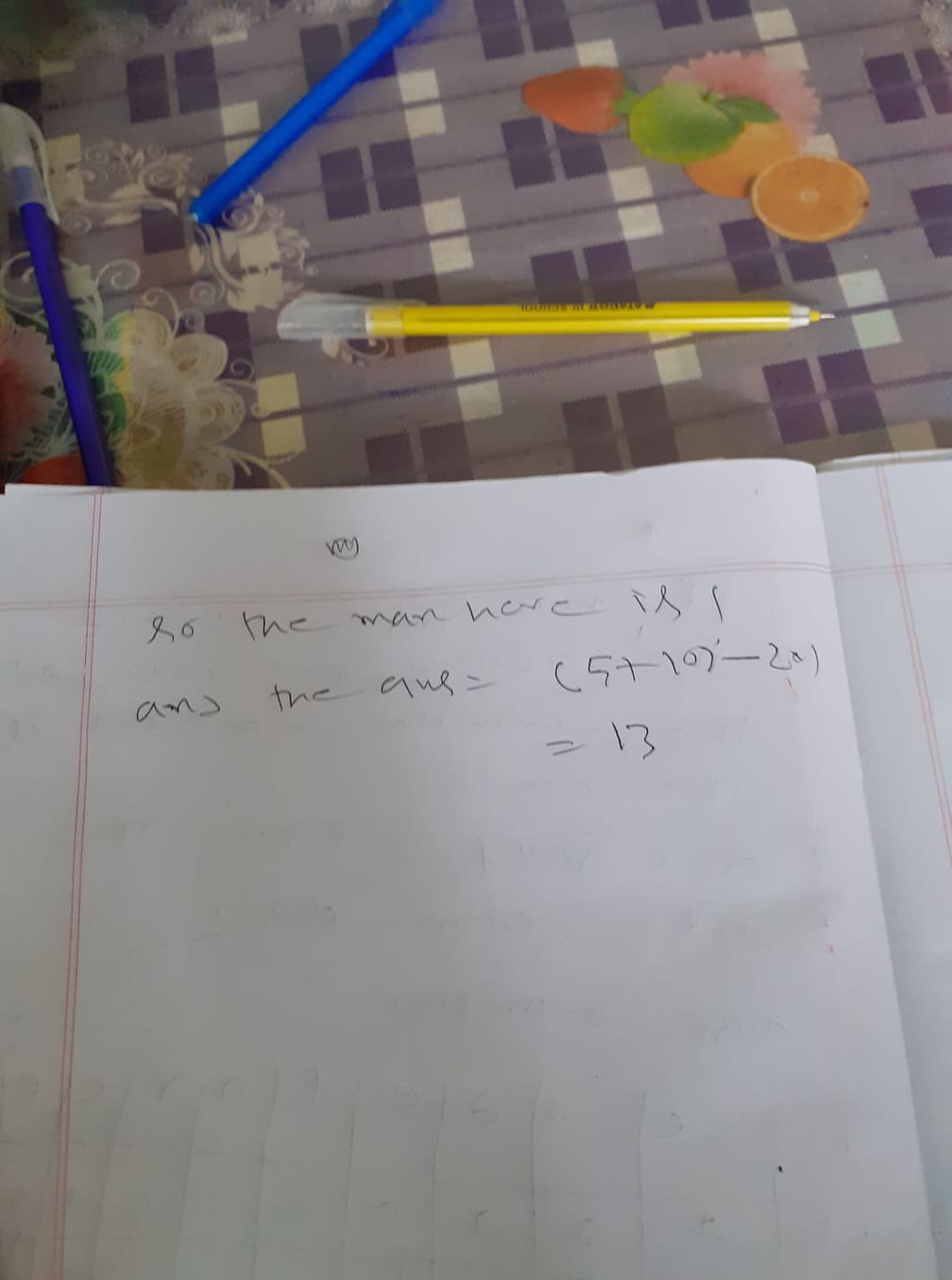
This is the problem statement:



The process of our solution is :

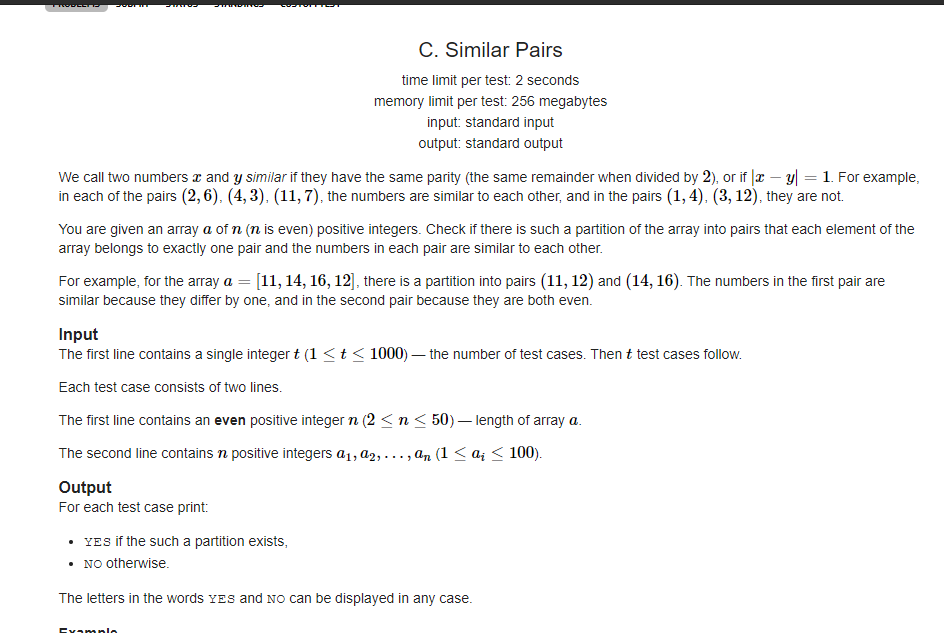




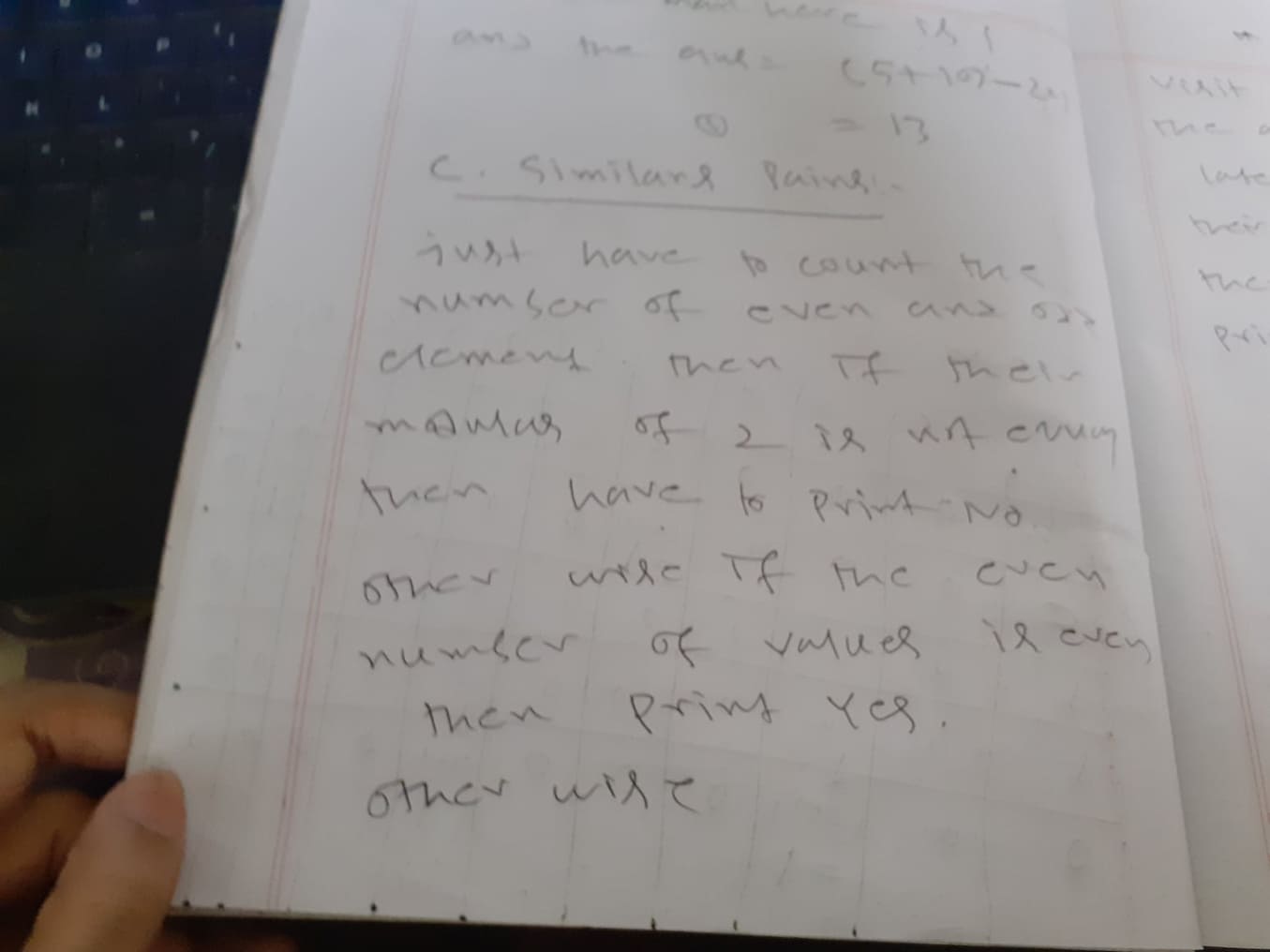


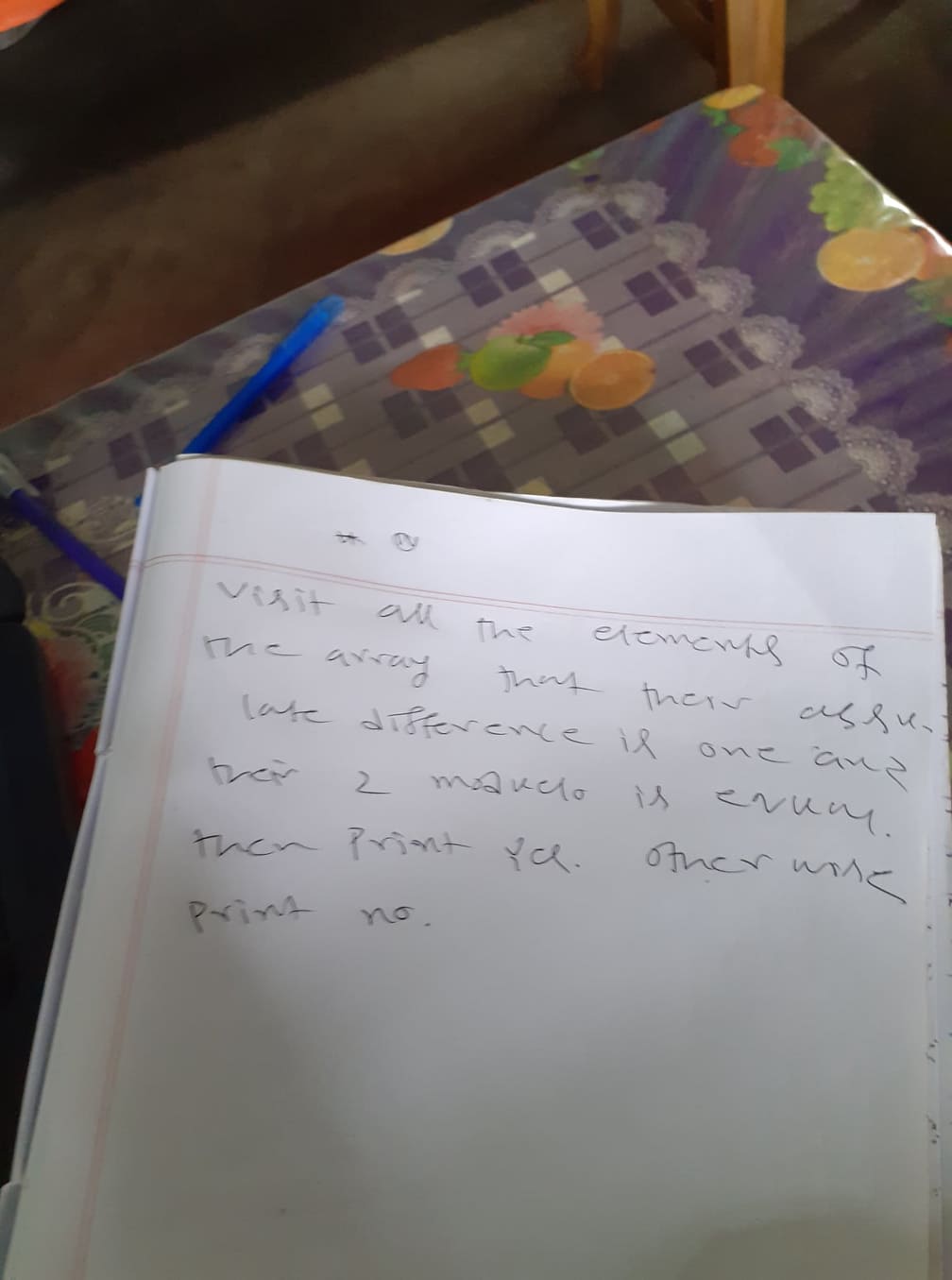
C. Similar Pairs(<https://codeforces.com/problemset/problem/1360/C>)

The problem statement is given below:



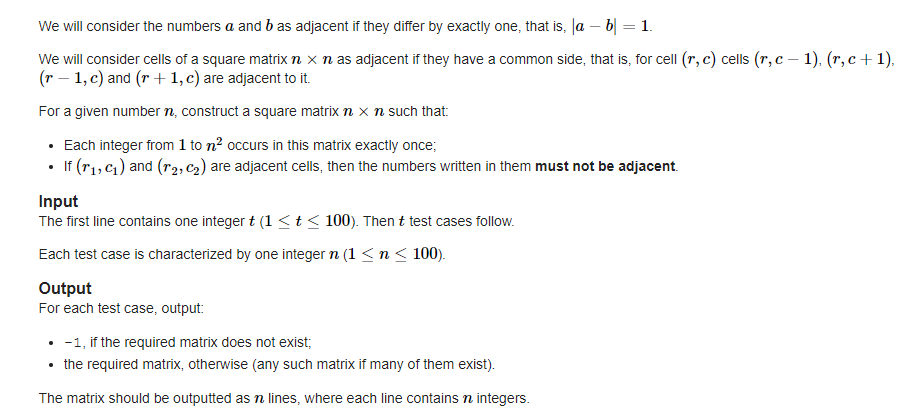
The problem solving process is given below:





C. Not Adjacent Matrix(<https://codeforces.com/problemset/problem/1520/C>)

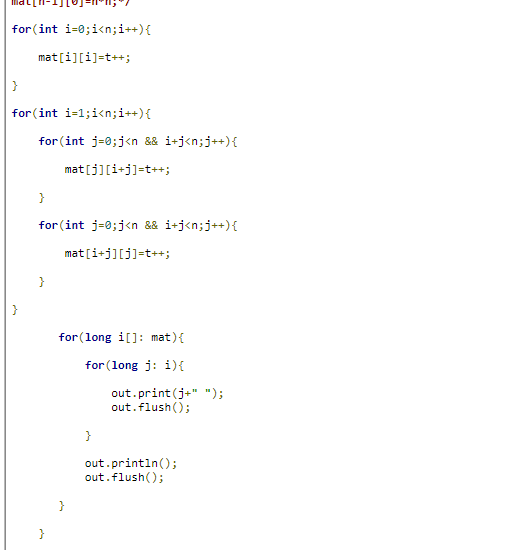
This is the problem statement:



We can consider the numbers a,b are adjacent if and only if their difference is 1.

Our task is make an matrix such that for (r,c) the all the side element are adjacent.

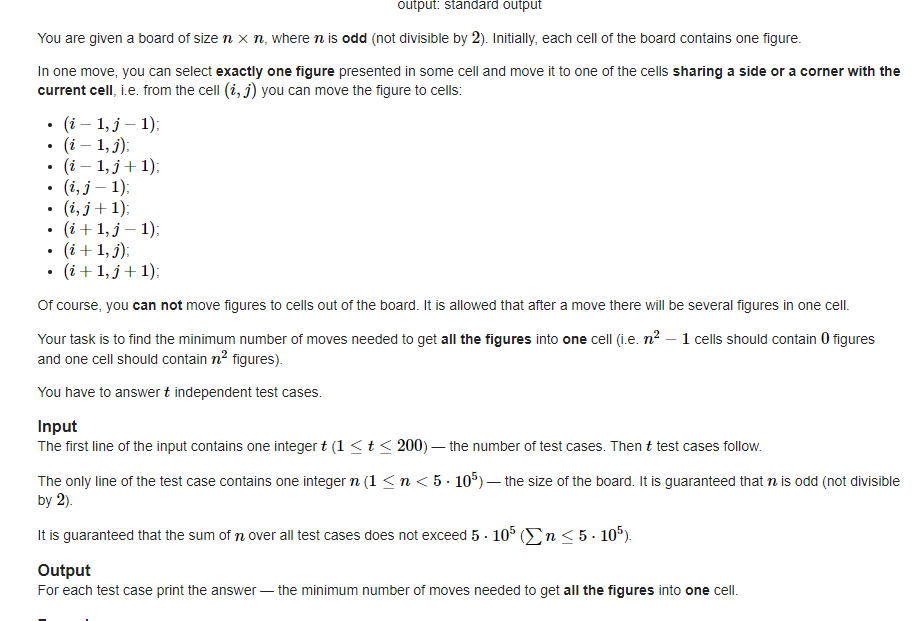
For this task we just use this algorithom to complete our task:



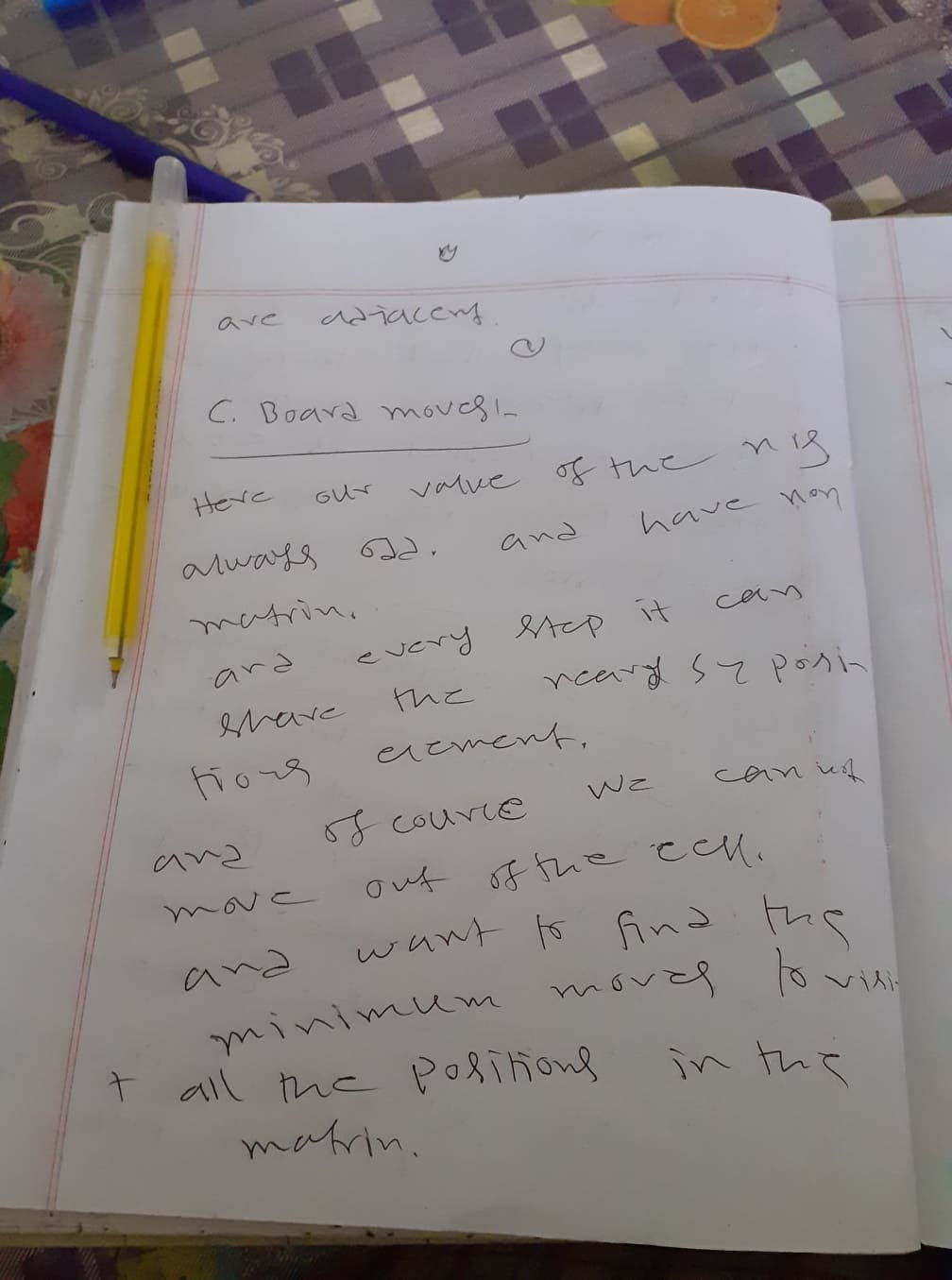
Where t start’s from 1.

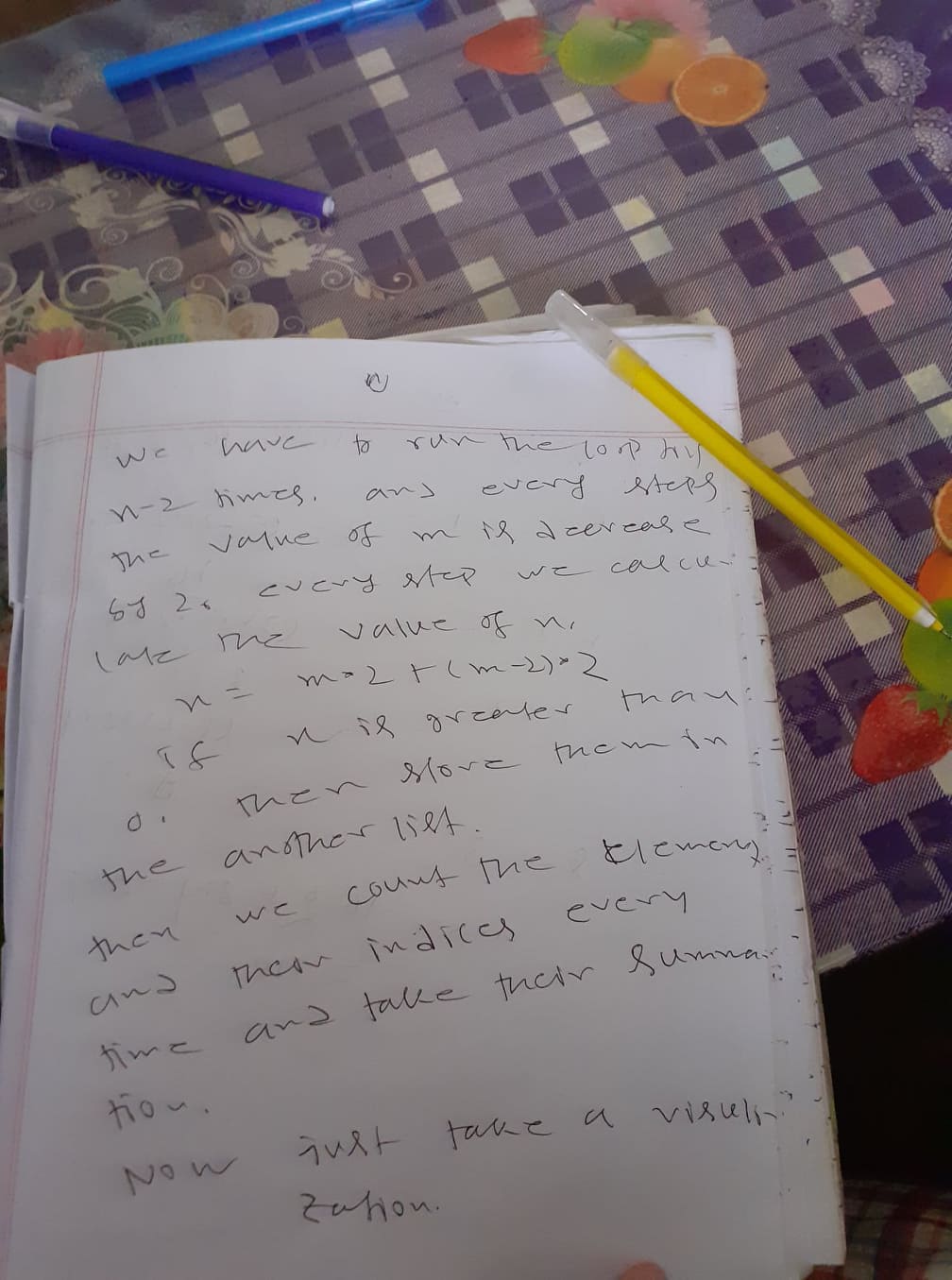
C. Board Moves(<https://codeforces.com/problemset/problem/1353/C>)

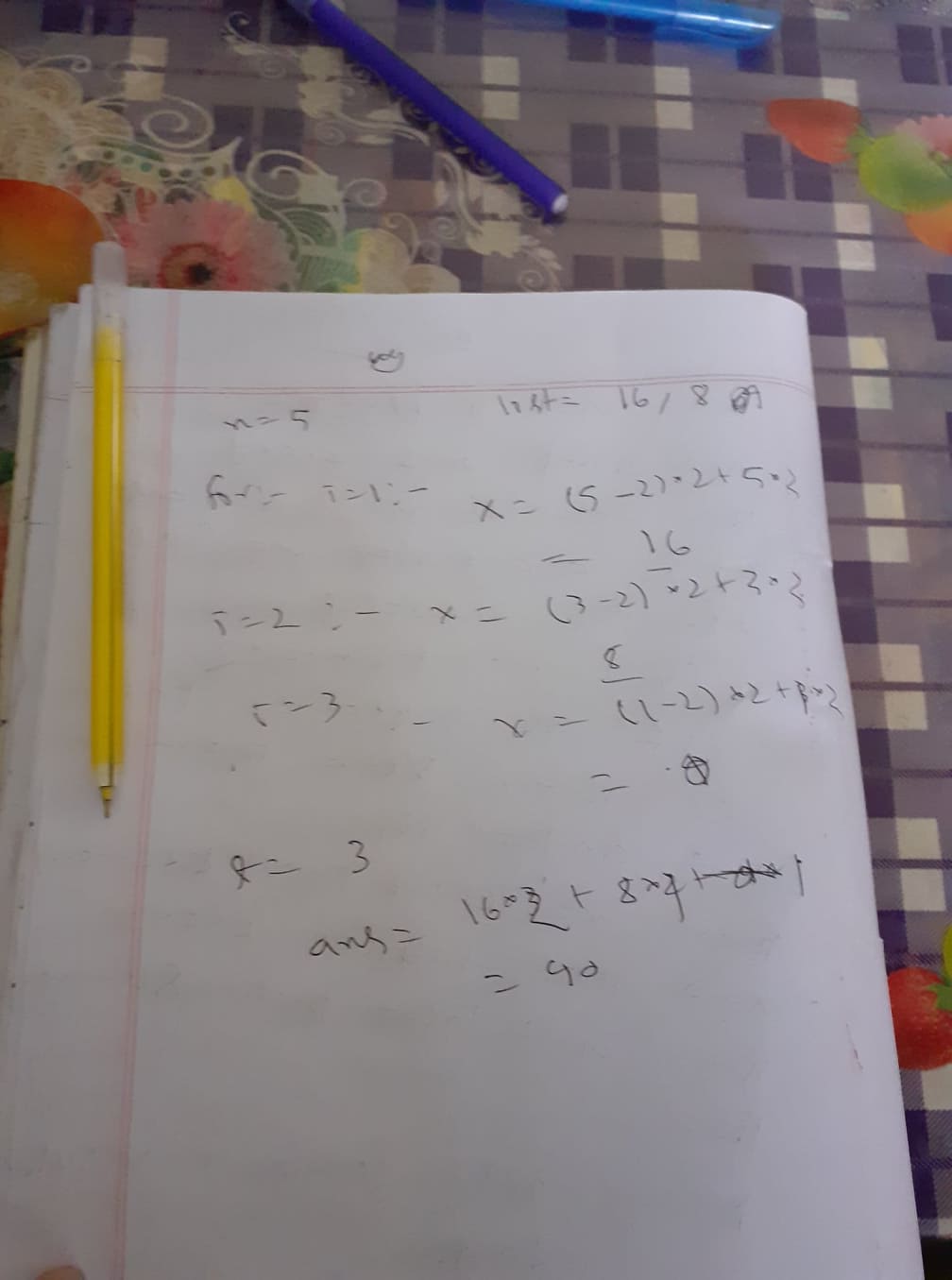
The problem statement is given below:



Our solution approach is given below:

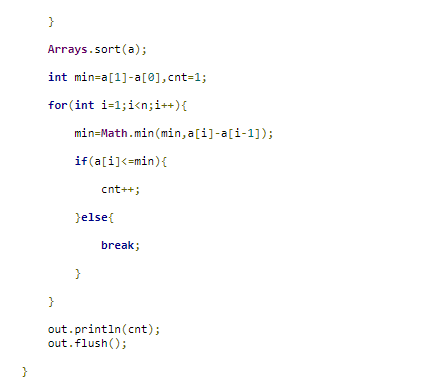






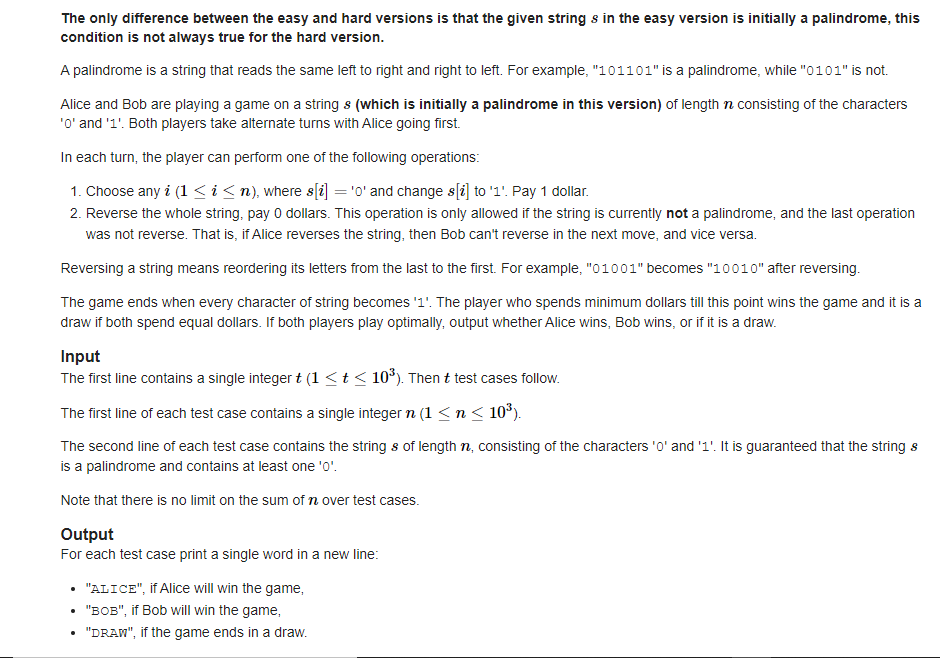
B. Sifid and Strange Subsequences(<https://codeforces.com/problemset/problem/1529/B>)

To solve this we just have to take a merge sort algorithom to sort our array. The have find the number of minimum adjacent element’s difference of the array. And this is the answer of our program.



B1. Palindrome Game (easy version)( <https://codeforces.com/problemset/problem/1527/B1>)

This is the problem statement



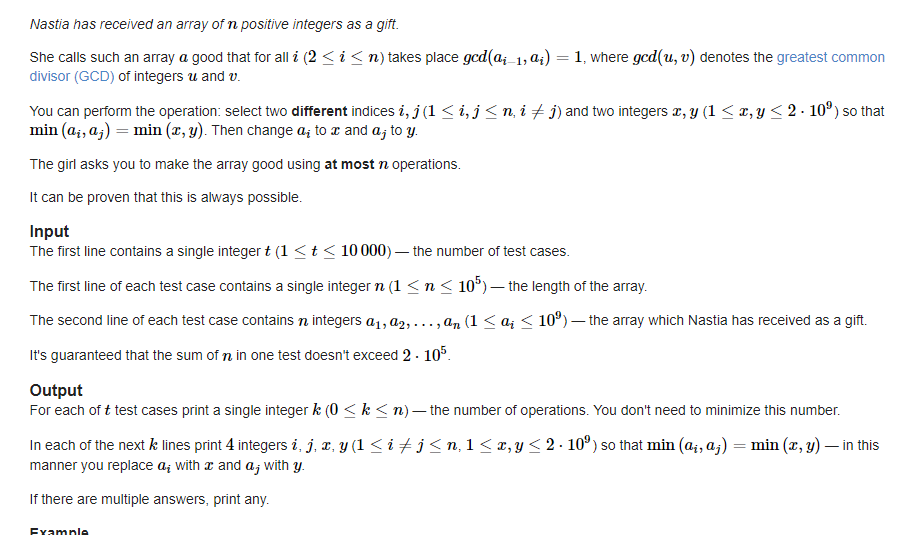
To solve this first we have to count the total number of zero’s the inputed string

Then if count is less than 2 then bob win, other wise if count is even then also bob win

Other wise half indexed element is 0 only then alice will win the game. Other wise bob will win the game.

B. Nastia and a Good Array(<https://codeforces.com/problemset/problem/1521/B>)

The problem statement is given below:



This is also the simple the problem. We have total n-1 number of process.

The we have to find the minimum element and it’s index

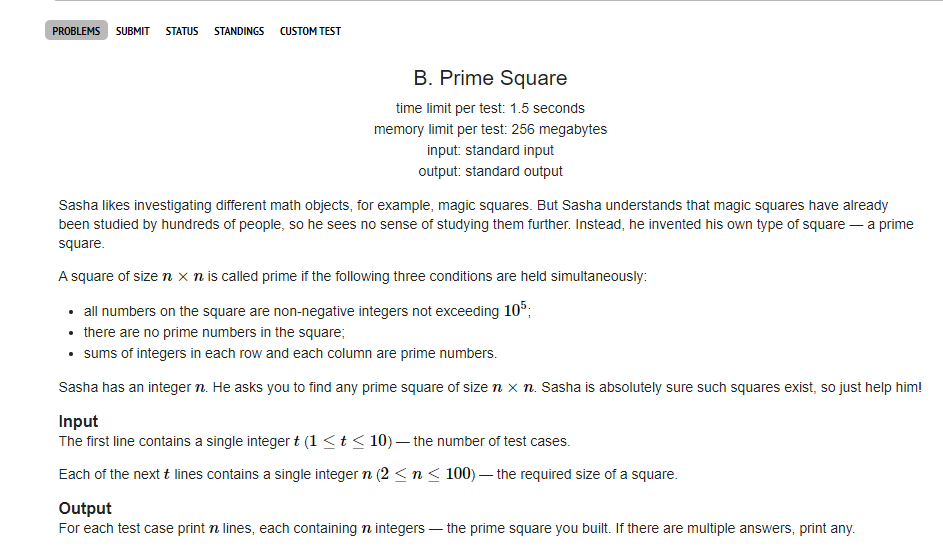
And without that minimum index at every state we just print the minimum element’s position

Current element position minimum value and minimum value and absolute difference of the minimum value’s positions and current element position

So the condition   
min(ai,aj)=min(x,y) is full filled

B.Prime Square(<https://codeforces.com/problemset/problem/1436/B>)

Here is our problem statement:



Here if n is even then we have to make the every coloumn’s first and last element =1

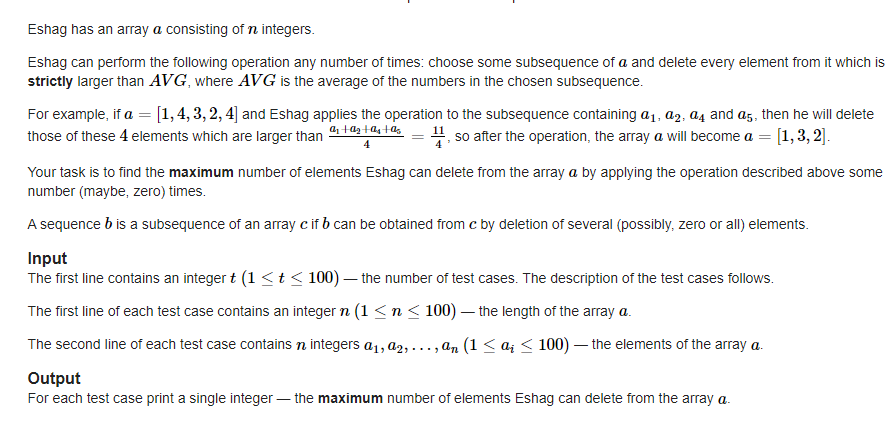
Otherwise if it is odd we have to also perform an operation’s we perform for the even n.

Now here we just have to a addition that is our middle raw element of the matrix next space and middle coloumn element and for reverse condion element’s will be 1.

So our server will solve this problem.

1. Eshag Loves Big Arrays(<https://codeforces.com/problemset/problem/1529/A>)

Here is our problem statement:



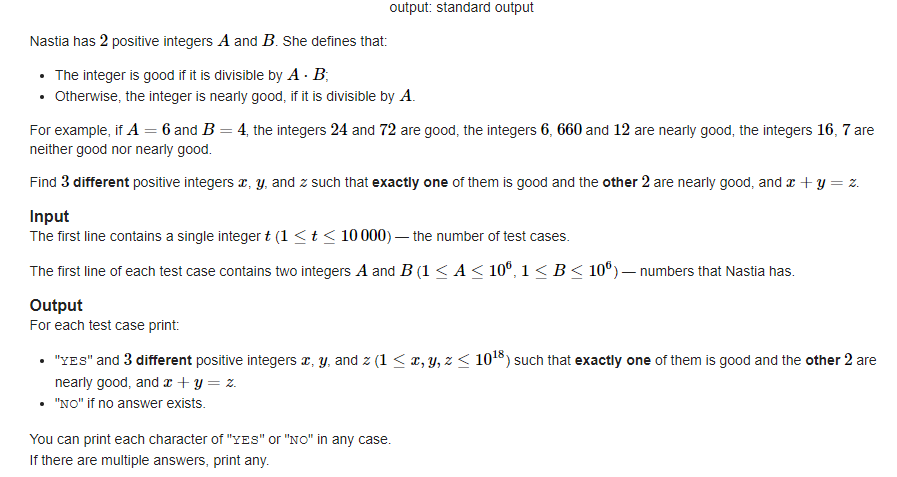
Here we have to find just the minimum element’s and how many time’s it present in the list.as the avg of the minimum value will be always minimum and for that reason we can remove the maximum number of element’s

Then our ans is n-k, where n is the of the array and k is number of time minimum value present in the array.

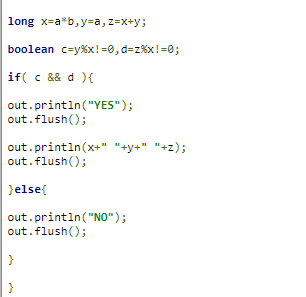
So our server also can solve this problem.

1. Nastia and Nearly Good Numbers(<https://codeforces.com/problemset/problem/1521/A>)

This is our problem statement’s:



Here we have to perform just a logical and operation.



And from this logic our problem will be solved.

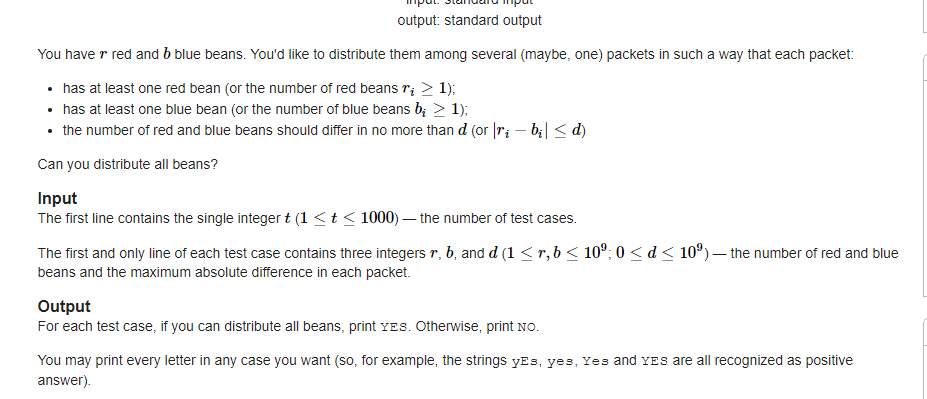
As there are There are 2 cases:

if B=1, then the answer doesn't exist. Here we cannot get the nearly good numbers at all.

Otherwise, we can construct the answer as A+A⋅B=A⋅(B+1).B

1. Red and Blue Beans(<https://codeforces.com/problemset/problem/1519/A>)

This is our problem statement:



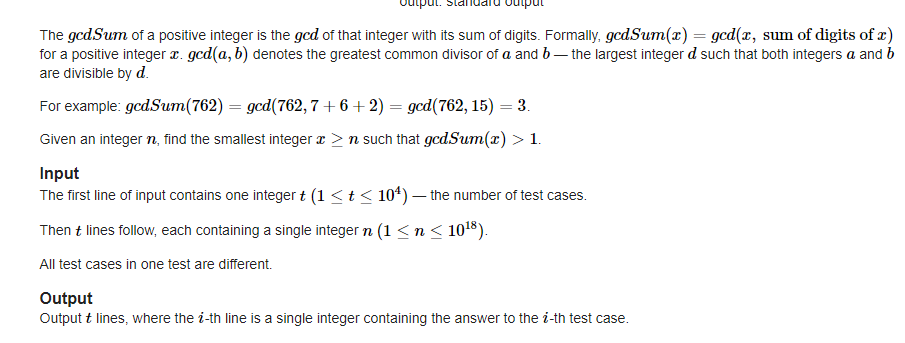
Our solution process is given below:

Without loss of generality, let's say r≤b (otherwise, we can swap them). Note that you can't use more than r packets (at least one red bean in each packet), so b can't exceed r⋅(d+1) (at most d+1 blue beans in each packet).

So, if b>r⋅(d+1) then asnwer is NO. Otherwise, we can form exactly r packets.

1. GCD Sum(<https://codeforces.com/problemset/problem/1498/A>)

This is our problem statement’s:



Here we have to use the large number of gcd’s algorithom to detect our gcd as our value is too large. The have to detect the summation of that large number’s digit.

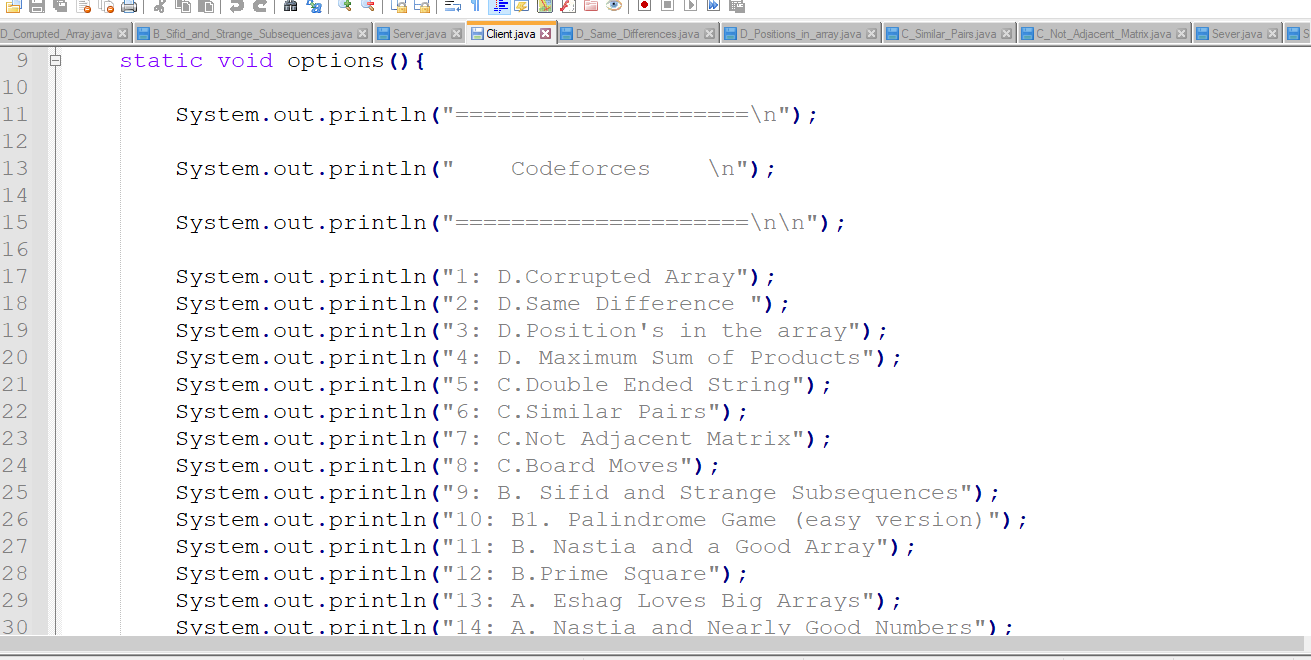
So if the summation of that digit’s and the orginal number’s gcd will be greater than 1, then print the value n. where n is the long number. Other wise increase tha vlae of n. run until we find the valid answer’s.

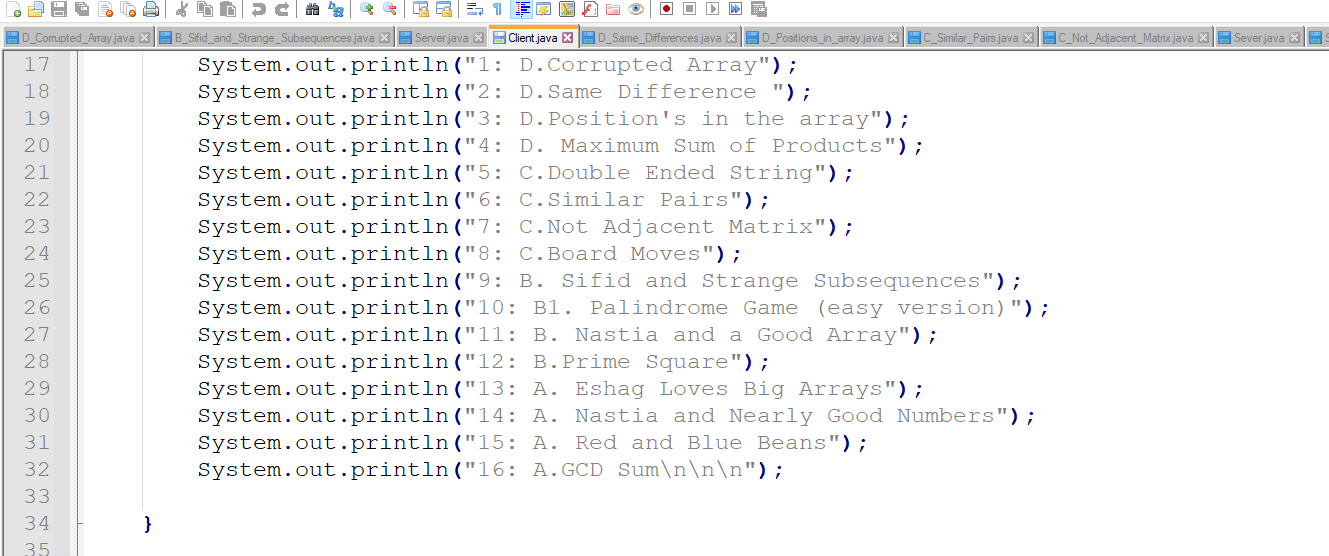
So this is the process of the problem’s here server client will discuss and they build algorithom and or build logic of them and finally make an acceble solution for the problem’s.

Now we discuss about the client;s task to send the problem code to server to solve that problem.

**Client**:

In the client clas clinet just chose the option’s he/she want to solve and send it to te server.





This is our option’s method.

And every time he chose the problem and send the problem code to the server as like as the same way we discuss in the previously.

When it wants to leave it types bye.

**File Passing:**

**Server class:** We created the environment to connect multiple sockets with the system. By creating socket variable client accepted the socket request. As CPU is the byte accessible so that we convert the data or information given in a file to byte array and then sent it to client.

**Client Class:** Client received the byte array that server sent to the client from its own console. Then it created an empty file & wrote the information from the byte array that accept from the server. And wrote its own console or its empty file.

=================================

Thank you

=====================================