Specialized Programming

Report No.#3



**Program: Master of Computer Science**

**Course: COMP SCI 7007 Specialised Programming PG**

**Group No.#5**

**Group Member:**

* **Pranshu Raj Goel (a1875750)**
* **Pratik Abhay Kurwalkar (a1867176)**
* **Ravi Trivedi (a1827896)**

Table of Contents

[Explain the rationale behind your choice of problems to attempt this week. 4](#_Toc111294955)

[List Classes/ Library Files and methods you used to solve each problem. 4](#_Toc111294956)

[o C++: 4](#_Toc111294957)

[ Library/ Header Files: 4](#_Toc111294958)

[ Functions Used: 4](#_Toc111294959)

[o Java: 4](#_Toc111294960)

[ Classes Used: 4](#_Toc111294961)

[ Functions Used: 5](#_Toc111294962)

[List the problems you attempted, your scores so far for each problem. 5](#_Toc111294963)

[1. HungryCowsEasy 5](#_Toc111294964)

[ Which steps of the problem-solving process did you find easy? 5](#_Toc111294965)

[ Which steps did you struggle with? 5](#_Toc111294966)

[ What approaches did you use to overcome those difficulties? 5](#_Toc111294967)

[ Can you think of how you might approach the problem differently if you had to start again? 5](#_Toc111294968)

[ Test Evidence: 6](#_Toc111294969)

[2. BearPair 7](#_Toc111294970)

[ Which steps of the problem-solving process did you find easy? 7](#_Toc111294971)

[ Which steps did you struggle with? 7](#_Toc111294972)

[ What approaches did you use to overcome those difficulties? 7](#_Toc111294973)

[ Can you think of how you might approach the problem differently if you had to start again? 7](#_Toc111294974)

[ Test Evidence: 7](#_Toc111294975)

[3. Sudoku Typo 9](#_Toc111294976)

[ Which steps of the problem-solving process did you find easy? 9](#_Toc111294977)

[ Which steps did you struggle with? 9](#_Toc111294978)

[ What approaches did you use to overcome those difficulties? 9](#_Toc111294979)

[ Can you think of how you might approach the problem differently if you had to start again? 9](#_Toc111294980)

[ Test Evidence: 9](#_Toc111294981)

[4. Game Show Total 11](#_Toc111294982)

[ Which steps of the problem-solving process did you find easy? 11](#_Toc111294983)

[ Which steps did you struggle with? 11](#_Toc111294984)

[ What approaches did you use to overcome those difficulties? 11](#_Toc111294985)

[ Can you think of how you might approach the problem differently if you had to start again? 11](#_Toc111294986)

[ Test Evidence: 11](#_Toc111294987)

[5. AqaAsadiNames 13](#_Toc111294988)

[ Which steps of the problem-solving process did you find easy? 13](#_Toc111294989)

[ Which steps did you struggle with? 13](#_Toc111294990)

[ What approaches did you use to overcome those difficulties? 13](#_Toc111294991)

[ Can you think of how you might approach the problem differently if you had to start again? 13](#_Toc111294992)

[ Test Evidence: 13](#_Toc111294993)

# Explain the rationale behind your choice of problems to attempt this week.

We choose to practice brute force problem for this week as we wanted to refamiliarize ourselves with IDEs and the languages, its features and various functions it provides.

# List Classes/ Library Files and methods you used to solve each problem.

## C++:

### Library/ Header Files:

* + 1. iostream
    2. vector
    3. unordered\_map

### Functions Used:

* + 1. vector:
       1. vector<type> variable\_name(size): creates a vector of size.
       2. push\_back(): To append the data to the end of the vector.
       3. begin(): Gives iterator pointing to first element of collection.
       4. end(): Gives iterator pointing to the end of collection(However, it is ahead of the last value of the Collection).
       5. rbegin(): Gives iterator pointing to last element of collection.
       6. pop\_back(): Pops the last element off the collection.
    2. unordered\_map:
       1. unordered\_map\_iterator->first: returns the key pointed by the iterator of the collection (key-value pair).
       2. unordered\_map\_iterator->second: returns the value pointed by the iterator of the collection (key-value pair).
       3. begin(): Gives iterator pointing to first element of collection.
       4. end(): Gives iterator pointing to the end of collection(However, it is ahead of the last value of the Collection).
    3. sort(iterator\_start, iterator\_end): To sort the collection based on ascending Order of its values for given iterator range.
    4. lower\_bound(iterator\_start, iterator\_end, key\_value): Provides iterator to the sorted collection between iterator\_start to iterator\_end both inclusive, where returned iterator points to the value equal to or smallest value greater than key\_value if present; else returns end of iterator.

## Java:

### Classes Used:

1. String – Used to create string objects to store values.
2. ArrayList – Used to iterate through the given test cases.

### Functions Used:

1. System.out.println() – Used to print a parameter passed to it.

# List the problems you attempted, your scores so far for each problem.

## HungryCowsEasy

* + Problem URL: <https://community.topcoder.com/stat?c=problem_statement&pm=15099>
  + Test Case URL: <https://community.topcoder.com/stat?c=problem_solution&cr=40732861&rd=17298&pm=15099>
  + Test Cases Passes: 25 / 25
  + Language: C++

### Which steps of the problem-solving process did you find easy?

Hashing the indexes of Cows and Barns with distance from origin being their key. This help by having faster recovery time for indexes relative to their distance from origin which are hashed. Complexity reduced from O(n) = n to O(n) = 1.

### Which steps did you struggle with?

Mutating nested C++ STL data structures with pointers and iterators.

### What approaches did you use to overcome those difficulties?

* Hashing
* Binary Search using lower\_bound()
* Making the complexity:
* sorting and hashing of cow data: n + nlogn
* sorting and hashing of barn data: m + mlogm
* finding closest barn with respect to cow: n

Total O(n): 2n + nlogn + mlogm

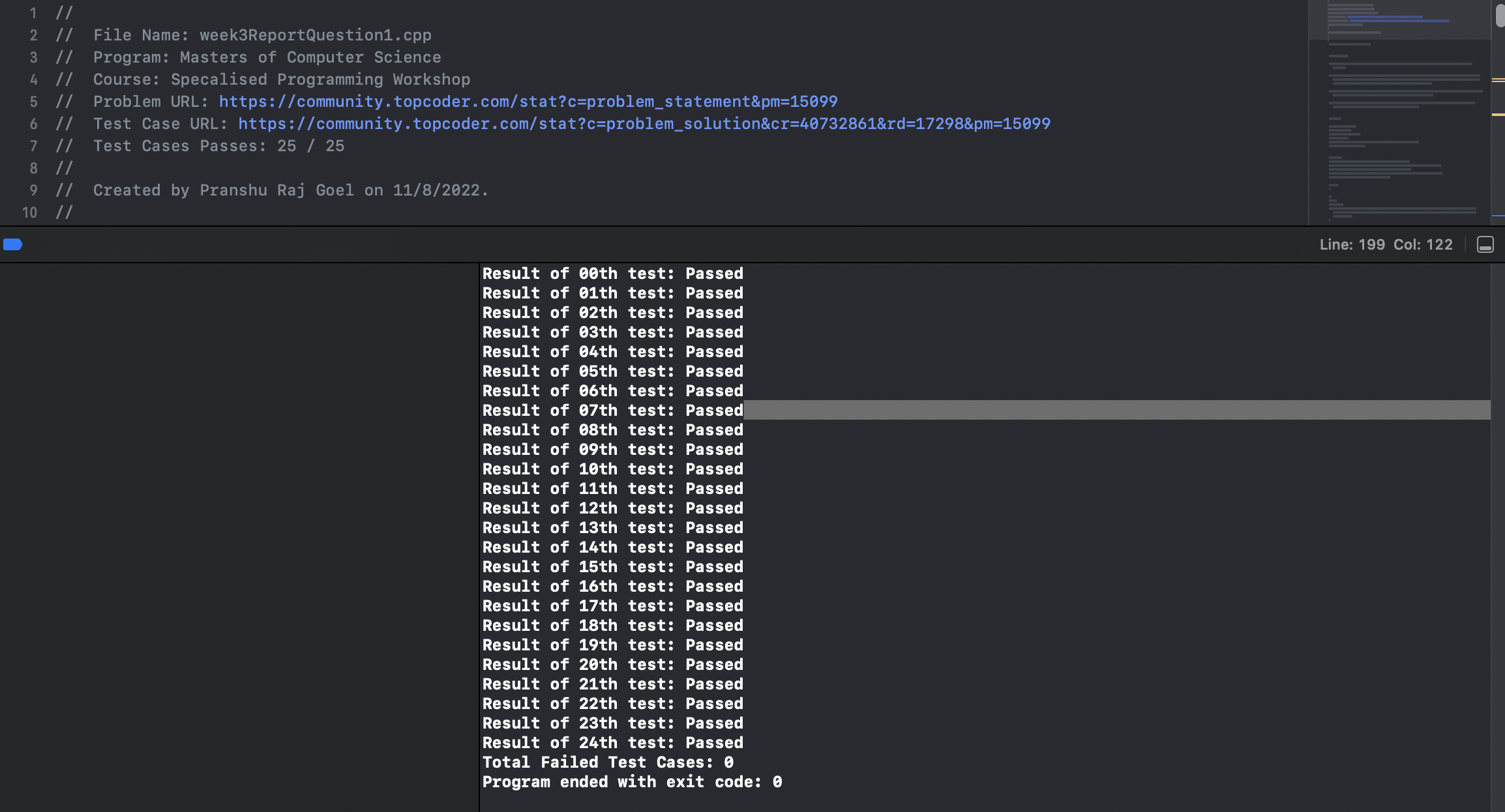
Instead of n\*m

### Can you think of how you might approach the problem differently if you had to start again?

If I had to start again,

* I would be more careful of the details provided, to ensure conflict are handled, if needed.
* Initialize size of collections if already known.
* Add a filter to utilize calculated barn info if more than 1 cow share a location.

### Test Evidence:



Source File: <https://github.com/pru-namikaze/CompetativeProgramming/blob/main/MasterOfComputerScience/SpecialisedProgramming/Report/Week3/week3ReportQuestion1.cpp>

## BearPair

* + Problem URL: <https://community.topcoder.com/stat?c=problem_statement&pm=14130>
  + Test Case URL: <https://community.topcoder.com/stat?c=problem_solution&cr=40346045&rd=16650&pm=14130>
  + Test Cases Passes: 56 / 56
  + Language: C++

### Which steps of the problem-solving process did you find easy?

### Which steps did you struggle with?

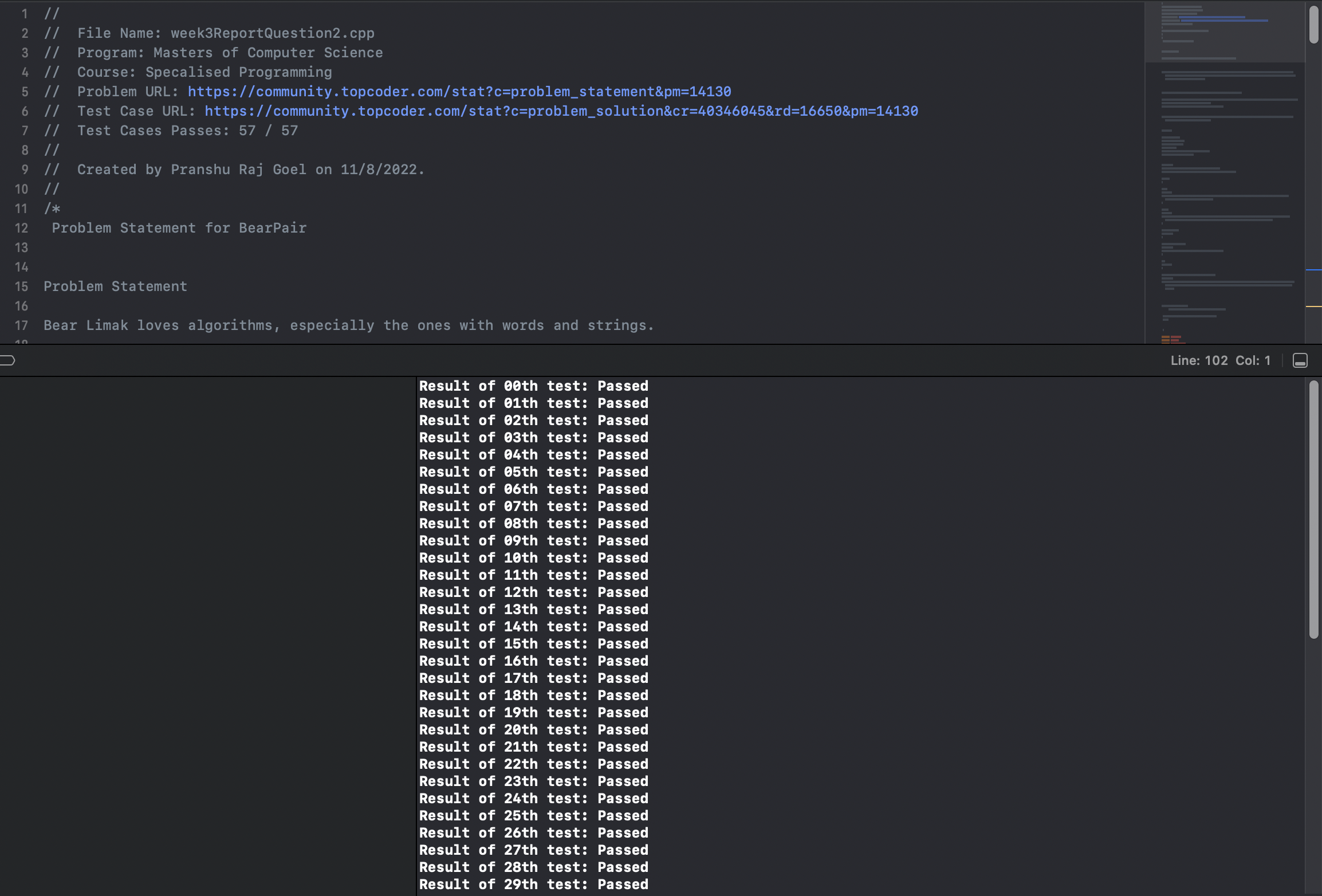
### What approaches did you use to overcome those difficulties?

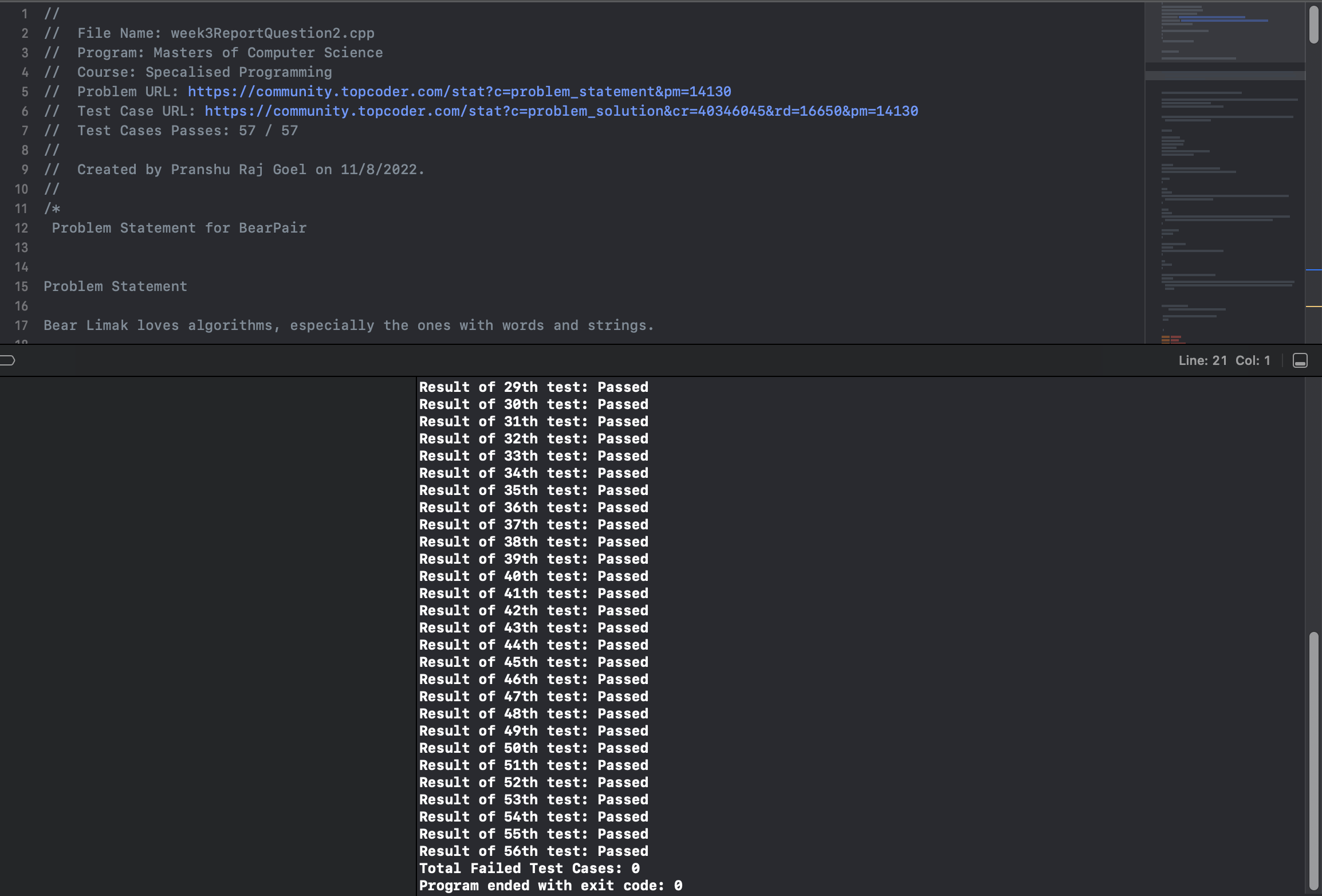
### Can you think of how you might approach the problem differently if you had to start again?

If I had to start again,

* Add a filter to utilize previously calculated fields calculated, as result for func(i, j) == func(j, i).

### Test Evidence:





Source File: <https://github.com/pru-namikaze/CompetativeProgramming/blob/main/MasterOfComputerScience/SpecialisedProgramming/Report/Week3/week3ReportQuestion2.cpp>

## Sudoku Typo

* + Problem URL: <https://community.topcoder.com/stat?c=problem_statement&pm=16988>
  + Test Case URL: <https://community.topcoder.com/stat?c=problem_solution&cr=15391415&rd=18686&pm=16988>
  + Test Cases Passes: 35/35
  + Language: Java

### Which steps of the problem-solving process did you find easy?

* I found that once I found an efficient way to get the required results, I was able to solve the problem using just 2 constant for loops.

### Which steps did you struggle with?

* After looking at the problem, at first, I tried to solve the problem by creating arrays of all the columns, rows, and 3 \* 3 matrix to check if each array is correctly solved.
* That method of solving was inefficient and not required for this type of problem.

### What approaches did you use to overcome those difficulties?

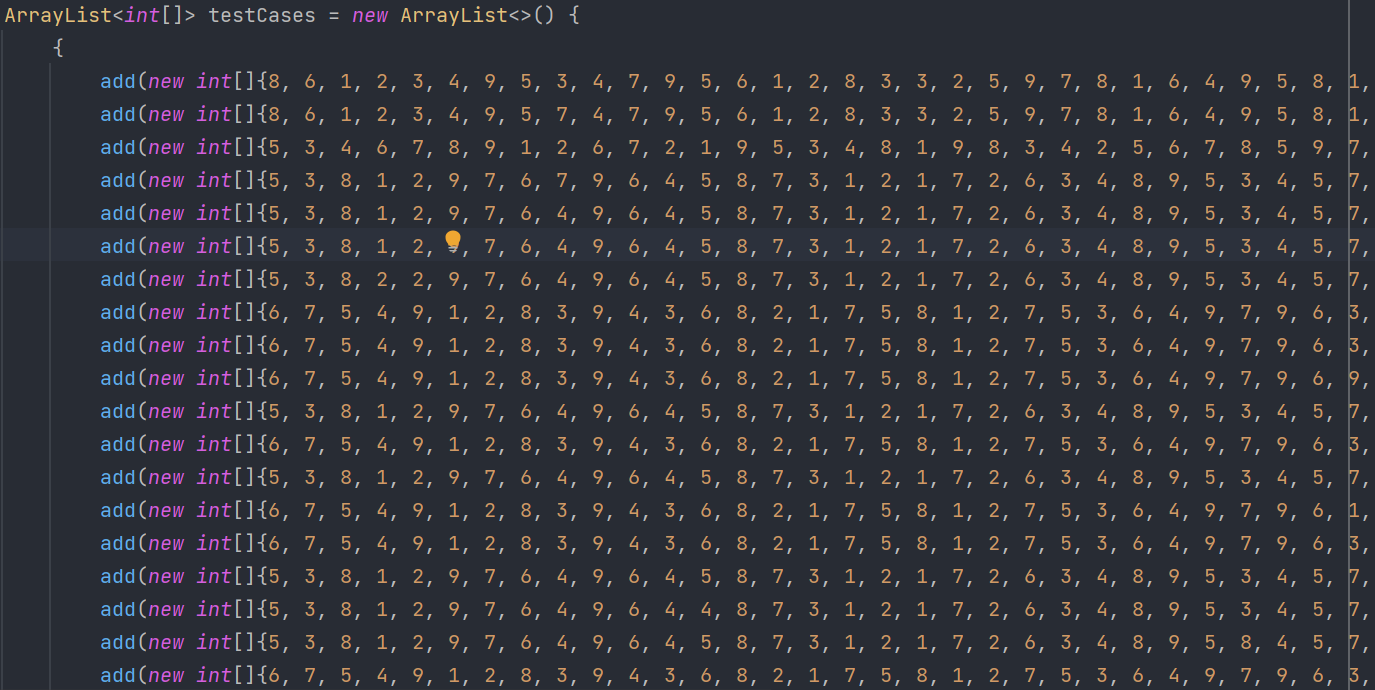
* I used a similar brute force approach to solve the problem with different implementation.

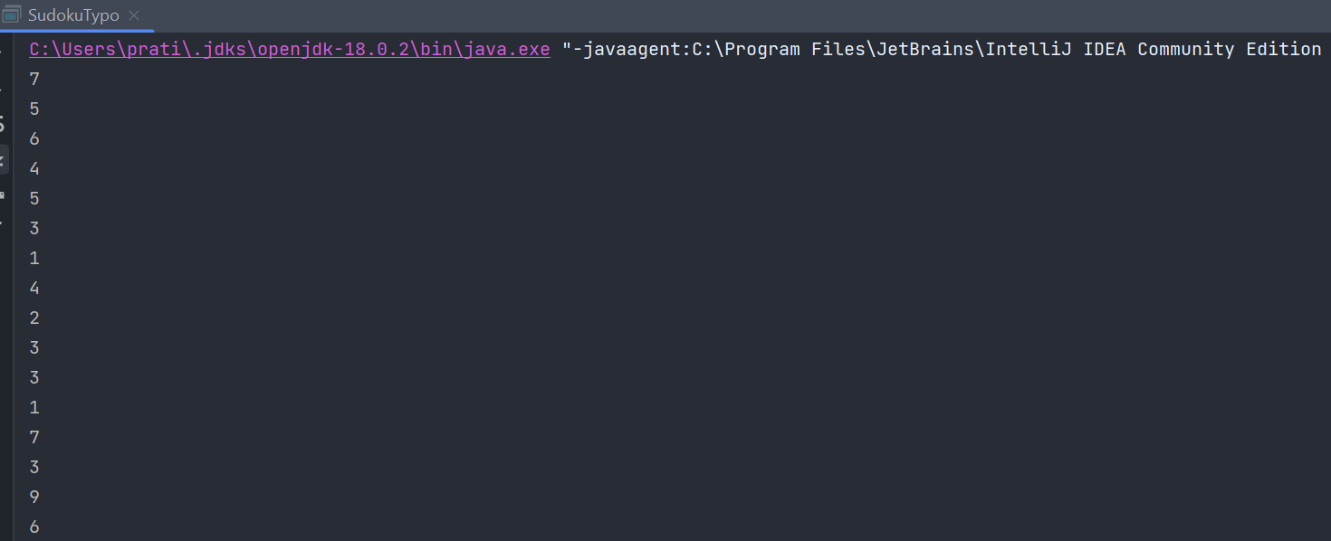
### Can you think of how you might approach the problem differently if you had to start again?

* If I had to start again, I would analyse the problem requirements completely and then chose the most efficient way to get the desired results.

### Test Evidence:

Test case results were compared with results available in topcoder.





Source File: <https://github.com/pratik-kurwalkar/SpecializedProgramming/blob/fb9a9c0ae3566f643727c0fc3b6c35a41ca62d06/SudokuTypo.java>

## Game Show Total

* + Problem URL: <https://community.topcoder.com/stat?c=problem_statement&pm=17590&rd=19218&rm=&cr=23082681>
  + Test Case URL: <https://community.topcoder.com/stat?c=problem_solution&cr=23082681&rd=19218&pm=17590>
  + Test Cases Passes: 25/25
  + Language: Java

### Which steps of the problem-solving process did you find easy?

### Which steps did you struggle with?

### What approaches did you use to overcome those difficulties?

### Can you think of how you might approach the problem differently if you had to start again?

If I had to start over and approach the problem again, would look for ways to reduce the time complexity of the problem. Currently, the time complexity of the problem is O(n2), However, there are ways of approaching the problem with linear time complexity.

### Test Evidence:

Test case results were compared with results available in topcoder.

Text

Description automatically generated

Text

Description automatically generated

Source File: <https://github.com/pratik-kurwalkar/SpecializedProgramming/blob/35214f7efa4ca2c107e7fa2eab6495eee57f8c3b/GameShowTotal.java>

## [AqaAsadiNames](https://community.topcoder.com/stat?c=problem_statement&pm=16195&rd=17994)

* + Problem URL: <https://community.topcoder.com/stat?c=problem_statement&pm=16195&rd=17994>
  + Test Case URL: <https://community.topcoder.com/stat?c=problem_solution&cr=40652696&rd=17994&pm=16195>
  + Test Cases Passes: 25/25
  + Language: Java

### Which steps of the problem-solving process did you find easy?

It was simple task to divide the strings into two parts with whitespace and according to the conditions use it accordingly.

### Which steps did you struggle with?

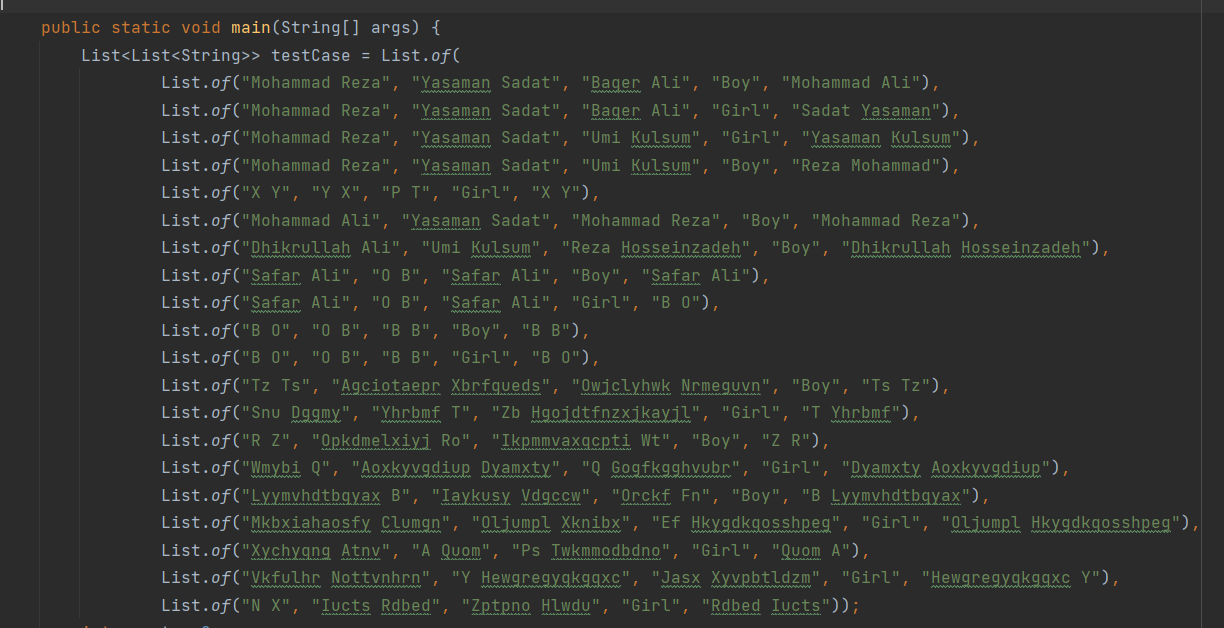
### What approaches did you use to overcome those difficulties?

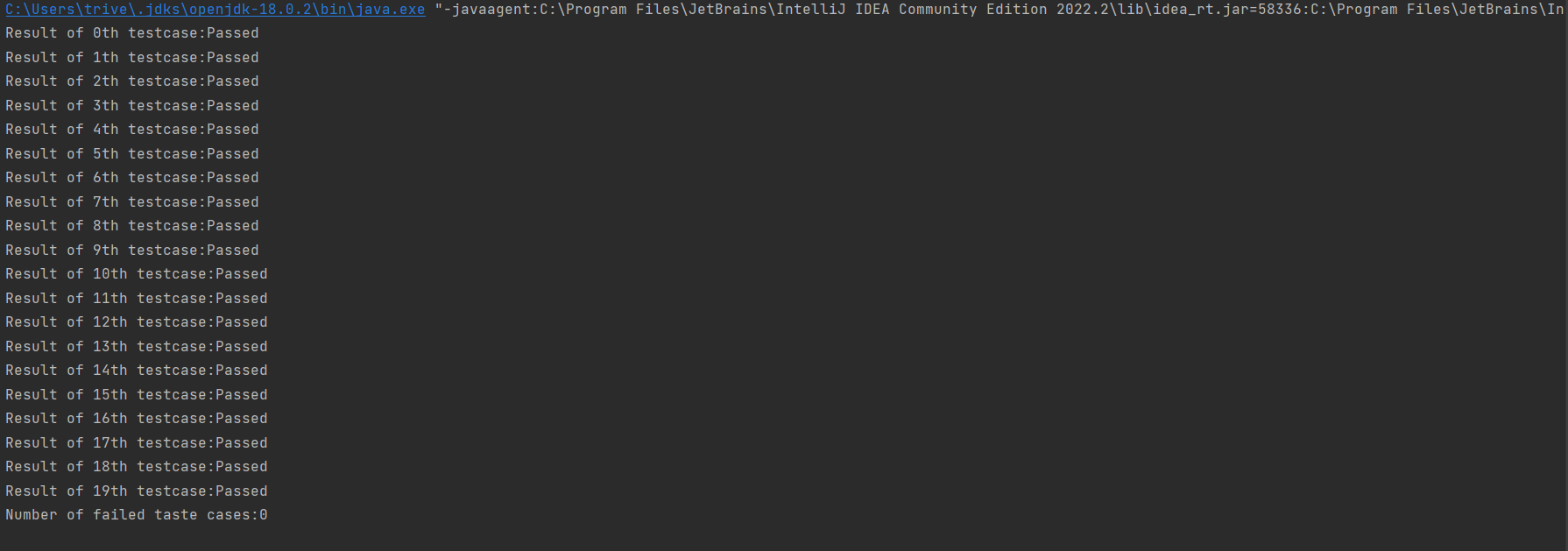
I used the simple String manipulation and in order to get the first name and second name of the second child.

### Can you think of how you might approach the problem differently if you had to start again?

### Test Evidence:

Test case results were compared with results available in topcoder.





Source File: <https://github.com/Ravi2409/Specialised-Programming/blob/main/AqaAsadiNames.java>