Exercise 6 (for grade) ~ Wednesday, October 26, 2022 ~ CPSC 535 Fall 2022

Write one submission for your entire group, and write all group members' names on that submission. Turn in your submission before the end of class. The X symbol marks where you should write answers.

Recall that our recommended problem-solving process is:

1. **Understand** the problem definition. What is the input? What is the output?
2. **Baseline** algorithm for comparison
3. **Goal** setting: improve on the baseline how?
4. **Design** a more sophisticated algorithm
5. **Inspiration** (if necessary) from patterns, bottleneck in the baseline algorithm, other algorithms
6. **Analyze** your solution; goal met? Trade-offs?

Follow this process for each of the following computational problems. For each problem, your submission should include:

a. State are the input variables and what are the output variables
b. Pseudocode for your baseline algorithm, that needs to include the data type and an explanation for any variable other than input and output variables
a. The Θ-notation time complexity of your baseline algorithm, with justification.

and if you manage to create an improved algorithm:

c. Answer the question: how is your improved algorithm different from your baseline; what did you change to make it faster?
d. Pseudocode for your improved algorithm, that needs to include the data type and an explanation for any variable other than input and output variables
a. The Θ-notation time complexity of your improved algorithm, with justification.

Today's problems are:

1.

Given the text T=Jimy_ran_and_hailed_the_monkey_to_stop and P=monkey, show how Horspool or Boyer-Moore algorithm works for searching the pattern P in the text T.

a) Construct the shift / bad shift table

b) Apply the Horspool or Boyer-Moore algorithm and show the shift after each unsuccessful match.

2.

Given the text T = t= 2359024411526739921 and the pattern P = 44115

(a) State the value of the prime q chosen to compute the hash values for the pattern and all the substrics T[s..s+2]

(b) Apply the Rabin-Karp algorithm to compute the indices where the pattern P exists in the text.

(c) State the number of valid matches and spurious hits in the Rabin-Karp algorithm

3.

Design an algorithm that, given an array ***S*** of ***n*** integers, identifies all integers that do not repeat, i.e. that show up only once in the array ***S***.

Your goal: baseline O(n^2) time, improved to either O(n log n) worst-case time or O(n) expected time.

Names

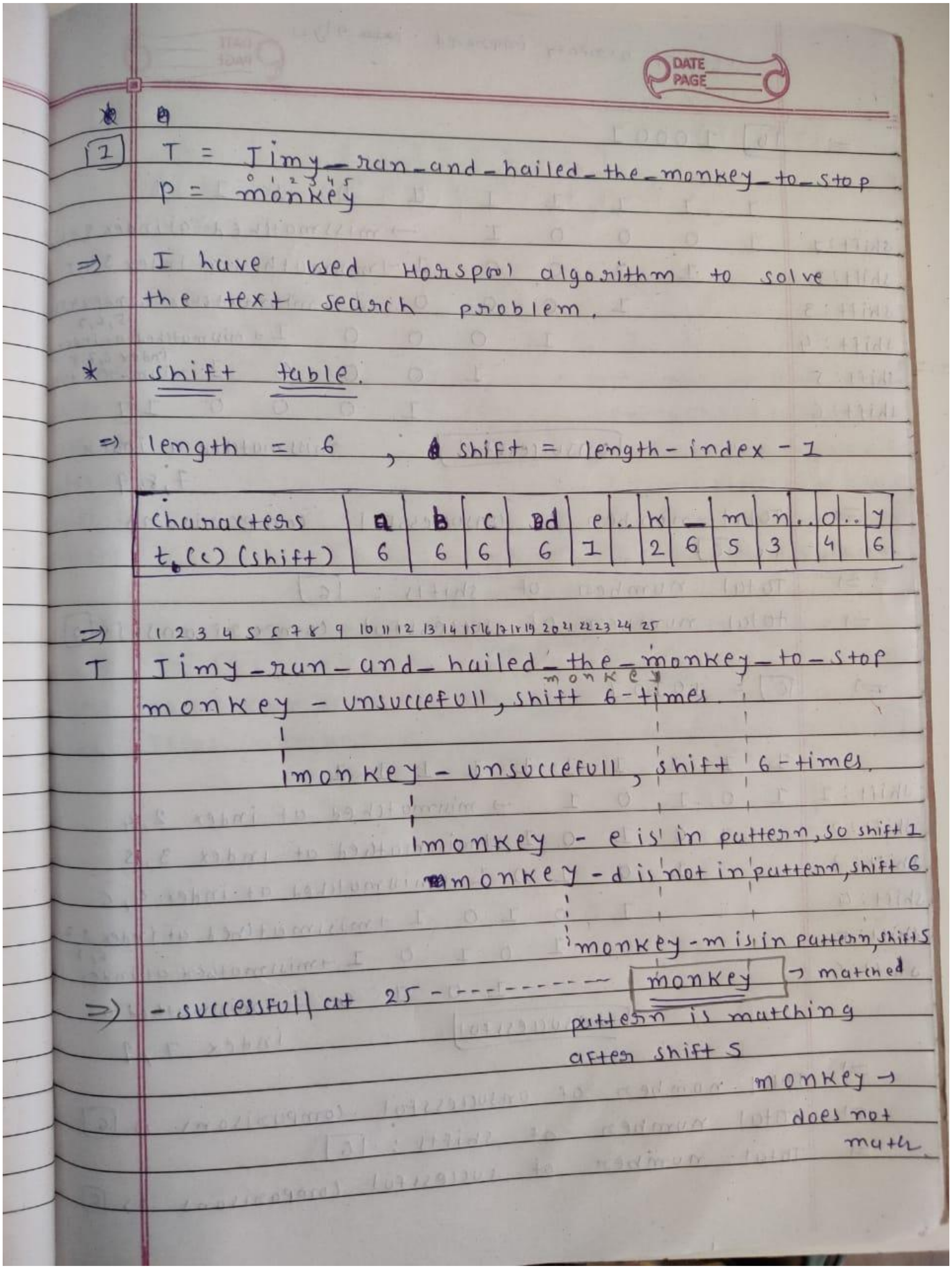
Write the names of all group members below.

☒ lency lakhani

Pradhatri Vemula

Hetal Patel

Exercise 1: Solve and provide answer



1) T = Jimy_ran_and_hailed_the_monkey_to_stop

p = monkey (0 1 2 3 4 5)

⇒ I have used Horspool algorithm to solve the text search problem.

* shift table.

⇒ length = 6 , shift = length - index - 1

characters	a	b	c	d	e	..	k	_	m	n	..	o	..	y
t(c) (shift)	6	6	6	6	1		2	6	5	3		4		6

⇒ 1 2 3 4 5 5 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

T Jimy_run_and_hailed_the_monkey_to_stop

monkey - unsuccefull, shift 6-times.

monkey - unsuccefull, shift 6-times.

monkey - e is in pattern, so shift 1

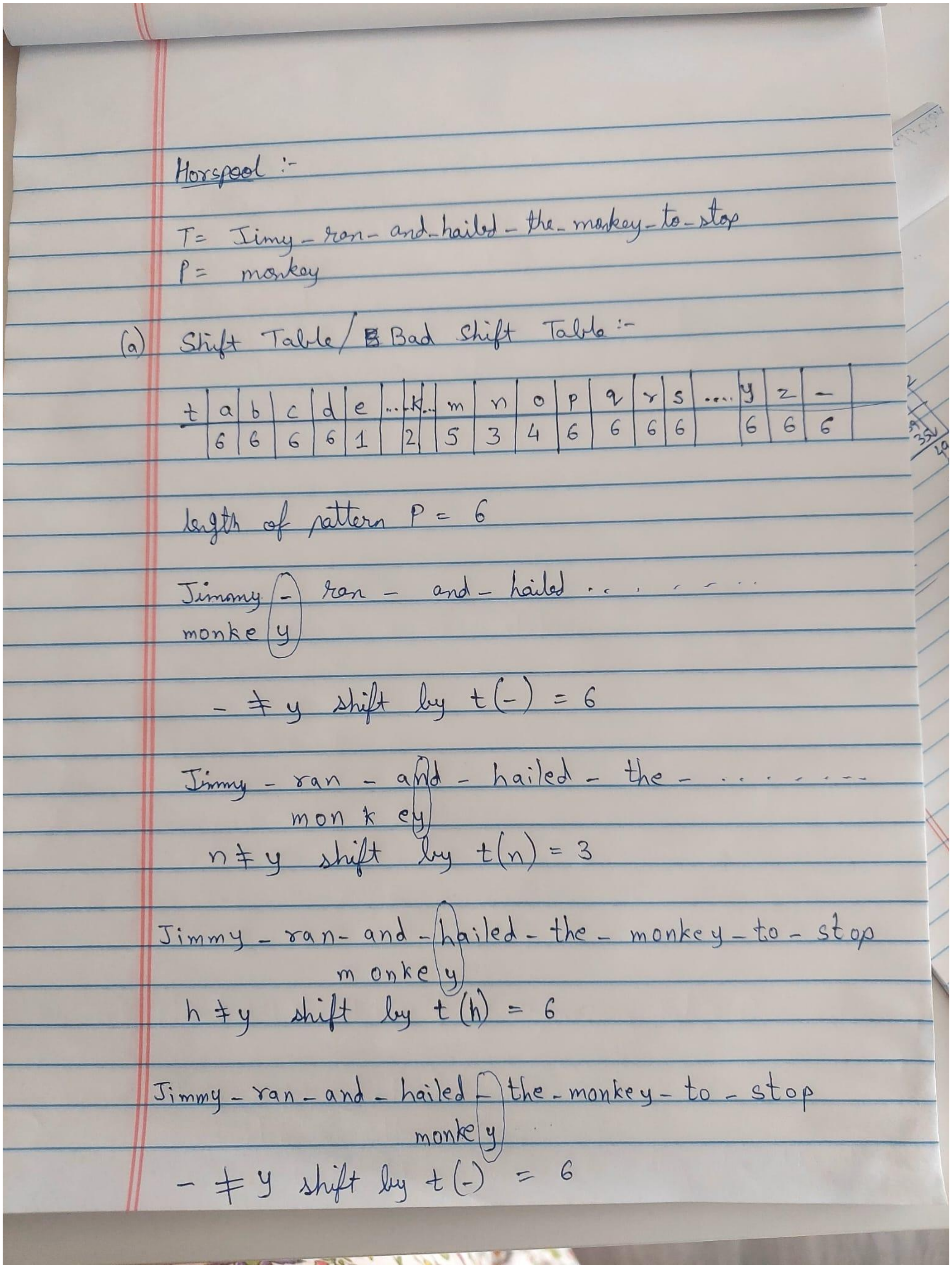
monkey - d is not in pattern, shift 6.

monkey - m is in pattern, shift 5

⇒ - successfull at 25 ---------- monkey → matched

pattern is matching after shift 5

monkey → does not math



Horspool :-

T= Jimy_ran_and_hailed_the_monkey_to_stop
P= monkey

(a) Shift Table/ ~~B~~ Bad Shift Table :-

t	a	b	c	d	e	...	k	...	m	n	o	p	q	r	s	....	y	z	_
	6	6	6	6	1		2		5	3	4	6	6	6	6		6	6	6

length of pattern P = 6

Jimmy _ ran _ and _ hailed . . . .
monke y

_ ≠ y shift by t(_) = 6

Jimmy _ ran _ and _ hailed _ the _ . . . . . .
mon k ey

n ≠ y shift by t(n) = 3

Jimmy _ ran _ and _ hailed _ the _ monkey _ to _ stop
m onke y

h ≠ y shift by t(h) = 6

Jimmy _ ran _ and _ hailed _ the _ monkey _ to _ stop
monke y

_ ≠ y shift by t(_) = 6

Jimmy_ran_and_hailed_the_monkey_to_stop
monkey

o ≠ y shift by t(o) = 4

Jimmy_ran_and_hailed_the_monkey_to_stop
monkey

m=m ; o=o ; n=n ; k=k ; e=e ; y=y ⇒ match

Horspool will terminate but if we keep going then,

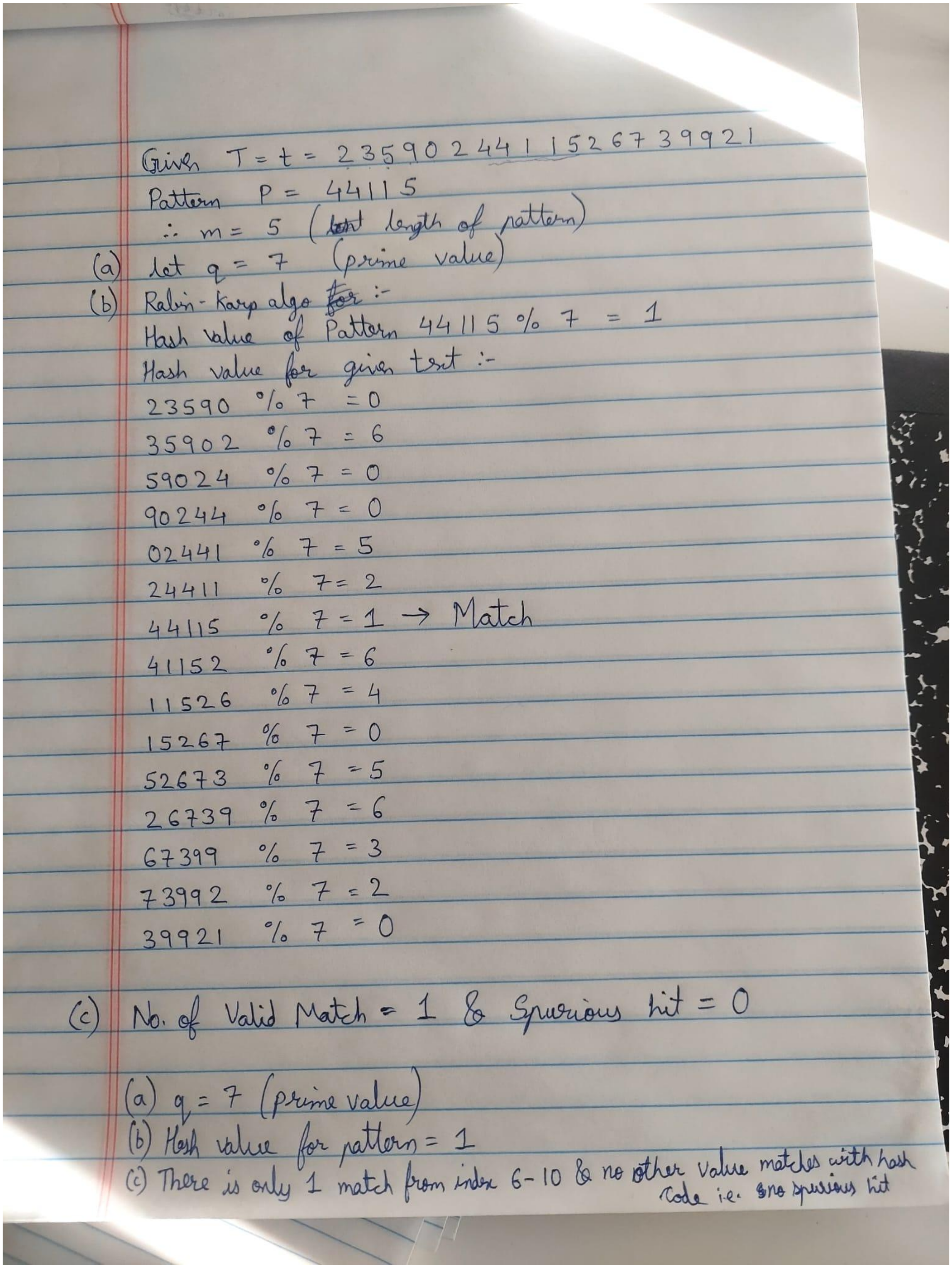
Jimmy_ran_and_hailed_the_monkey_to_stop
monkey

t ≠ y shift by t(t) = 6

∴ no other match found

Exercise 2: Solve and provide answer

X



Given T = t = 2359024411526739921

Pattern P = 44115

∴ m = 5 (~~text~~ length of pattern)

(a) let q = 7 (prime value)

(b) Rabin-Karp algo ~~for~~ :-

Hash value of Pattern 44115 % 7 = 1

Hash value for given text :-

23590 % 7 = 0
35902 % 7 = 6
59024 % 7 = 0
90244 % 7 = 0
02441 % 7 = 5
24411 % 7 = 2
44115 % 7 = 1 → Match
41152 % 7 = 6
11526 % 7 = 4
15267 % 7 = 0
52673 % 7 = 5
26739 % 7 = 6
67399 % 7 = 3
73992 % 7 = 2
39921 % 7 = 0

(c) No. of Valid Match = 1 & Spurious hit = 0

(a) q = 7 (prime value)
(b) Hash value for pattern = 1
(c) There is only 1 match from index 6-10 & no other value matches with hash code i.e. no spurious hit

Exercise 3: Solve and provide answer

Understand

Input/Output

Read and discuss the problem statement, then answer: **Describe the input and output definition in your own words.**

Input : is a given Array S with n integers

Output: all integers without duplicate values

Examples

Write out a concrete input value for the problem, and the concrete correct output. Repeat this at least once (so there are at least two input/outputs).

Given an array int numRay[] = { 0, 4, 3, 2, 7, 8,8, 2, 3,1,7,7,7};

The values to be returned are - 0,4,1

Given an array int numRay[] = { 0, 4, 3, 2, 7, 8, 3,1,7,7,7};

The values to be returned are - 0,4,2,8,1

Corner Cases

A corner case is a special case of input that is particularly unusual or challenging. Are there corner cases for this problem? If so, describe them.

There are no corner cases for this problem.

Use Case

Think of a specific use case for an algorithm that solves this problem. What kind of feature could it support in practical real-world software?

When sending gifts to people who had filled the forms on multiple portals. If we find out the list with non repeated entries, then the gifts can be sent only once.

Similar Problems

Does this problem resemble any other problems you remember? If so, which problem? Does that problem have an efficient algorithm? If so, what is the run time of that algorithm, inΘ-notation?

We can write a better algorithm that is efficient than the baseline. We can use data structures such as hashmaps to find out the duplicates more effectively. alternatively , we can use a part of Radix sort if the given elements are within the range of 0-k

Baseline

Design Overview

Design a naïve algorithm that solves the problem. This should be a simple algorithm that is easy to describe and understand. First, describe the basic idea of your algorithm in 1-2 sentences.

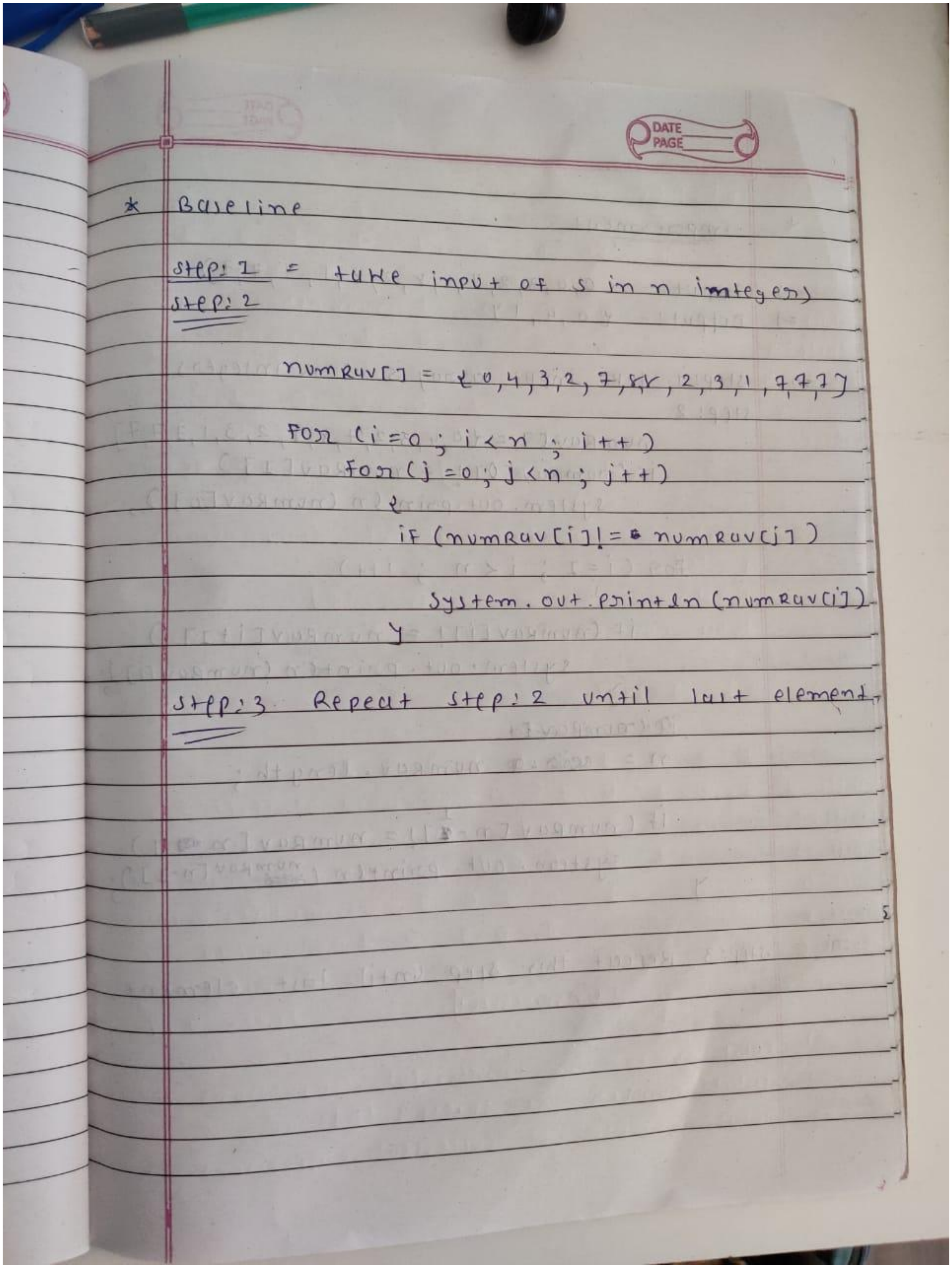
☒ First we store all the elements in the array in a list. Then we write two loops to loop through the array and find the duplicate element. Once a duplicate element is found, then all such elements are removed from the list

Pseudocode

Now, write clear pseudocode for your baseline algorithm.

```
NoDuplicates(int array)
    for i<-array.length
        list.add(array[i])
    for i<-array.length
        for j<-array.length
            If array[i]==array[j]
                Then list.removeAll(array[i])
    Return list
```

☒



* Baseline

step: 1 = take input of s in n integers)

step: 2

```
numRuv[] = {0,4,3,2,7,8,5,2,3,1,7,7,7]

for (i=0 ; i<n ; i++)
    for (j =0 ; j<n ; j++)
    {
        if (numRuv[i]!= numRuv[j])

            System.out.println(numRuv[i])
    }
```

step: 3 Repeat step: 2 until last element.

Efficiency

What is the Θ-notation time complexity of your baseline algorithm? Justify your answer.

O(n2) is the time complexity of the above approach

X

Improvement

Now, steps 3-6 involving creating a better algorithm. Only attempt this part if you finished the earlier parts and still have time.

Goal

What aspect of your baseline algorithm do you want to improve? Time efficiency, or something else?

To improve time efficiency of the baseline algorithm

X

Design Overview

Design a better algorithm for the same problem. First, describe the basic idea of your algorithm in 1-2 sentences.

Loop through all the elements in the array

Increase the arr[i]%n'th element by n for each element in the array.

Return to the array and display all indexes i for which arr[i]/n is larger than 1. This ensures that the number n has been added to the index.

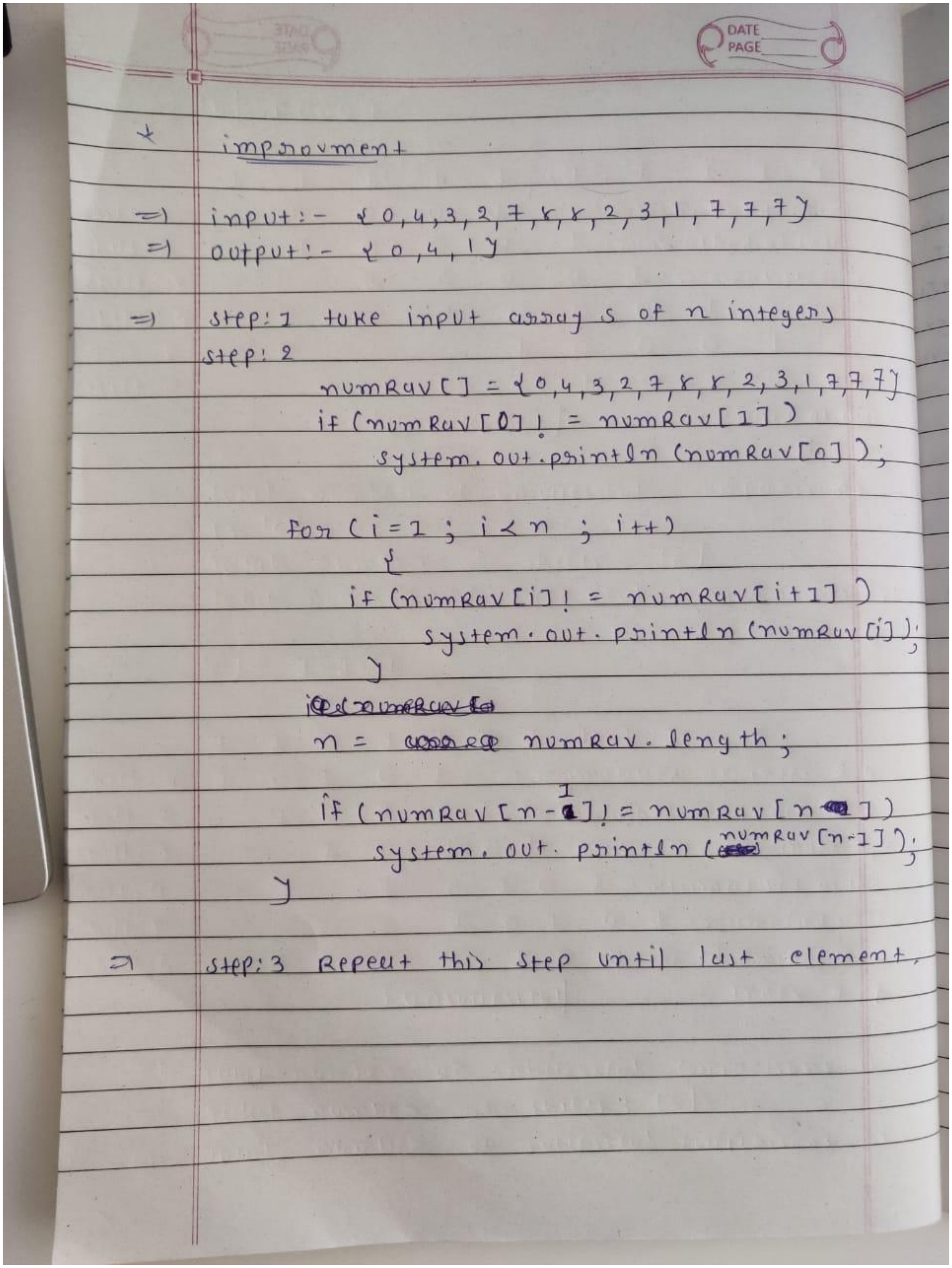
This method works because all elements are between 0 and n-1, and arr[i] would be greater than n only if the value I appeared more than once.

X

Pseudocode

Now, write clear pseudocode for your improved algorithm.

```
NoDuplicates(int array)
    for i<-array.length
            list.add(array[i])
    for i<-array.length
            array[array[i]%array.length]=array[array[i]%array.length]+array.length;
    for i<-array.length
            if(array[i]>=array.length*2)
                    then list.removeAll(i)
    return list
```



* improvment

=) input:- {0,4,3,2,7,5,5,2,3,1,7,7,7}

=) output:- {0,4,1}

=) step: 1 take input array s of n integers

step: 2

```
    numRav[] = {0,4,3,2,7,5,5,2,3,1,7,7,7}
    if (numRav[0]! = numRav[1])
        system.out.println (numRav[0]);

  for (i=1 ; i<n ; i++)
      {
      if (numRav[i]! = numRav[i+1] )
          system.out.println (numRav[i]);
      }
    n = numRav.length;

    if (numRav[n-1]! = numRav[n])
        system.out. println (numRav[n-1]);
  }
```

=) step: 3 Repeat this step until last element.

☒

Efficiency

What is the Θ-notation time complexity of your improved algorithm? Justify your answer.

The time complexity for the above pseudo code is O(n)

☒improvment efficiency is o(nlogn).

Analysis

Did you meet your goal? Why or why not?

The Goal has been met

☒