Algorithms and Data Structures (ADS) - COMP1819

Develop and optimise solutions in Python with ADS and provide complexity analysis.

Group Number: 3

Team members:

Member	Name
1	Hernandez Upegui, Bryan
2	Niazi, Nikmal
3	Luz, Louis Anthony
4	Saleh, Abdalla
5	Alpoim ferreira do rosario, Alana taiana

Contents

1. Create unique solutions!

Student 1-6's solution

Explain your understanding of the problem, and approach to solve it.

Short description and highlights of the **difference in your code**, and the full code in Appendix.

Student 1: Bryan's Solutio

```
1. import time
2.
3. # Function to check if a number is prime
4. def is_prime(n):
5.
6.     if n < 2:
7.         return False
8.     for i in range(2, int(n**0.5) + 1):
9.         if n % i == 0:
10.            return False
11.    return True
12.
13. # Function to check if a number is a palindrome
14. def is_palindrome(n):
15.
16.    return str(n) == str(n)[::-1]
17.
18. # Function to find all 'special numbers' that are both prime and palindromic
    within a range
19. def find_special_numbers(m, n):
20.
21.    special_numbers = []
22.    for num in range(m, n + 1):
23.        if is_prime(num) and is_palindrome(num):
24.            special_numbers.append(num)
25.    return special_numbers
26.
27.  # Function to display the 'special numbers' found. If more than five, show the
    first and last three
28. def display_special_numbers(special_numbers):
29.    total = len(special_numbers)
30.    if total < 6:
31.        print("Total: Special Numbers =", total)
32.        print("List of special numbers =", special_numbers)
33.    else:
34.        print("Total: special Numbers =", total)
35.        print("List of special numbers =", special_numbers[:3] +
    special_numbers[-3:])
36.
37. # Main function to drive the program
38. def main():

39.    m = int(input("Enter the lower limit (m): ")) # Prompt user for lower range
    limit.
40.    n = int(input("Enter the upper limit (n): ")) # Prompt user for upper range
    limit.
```

```
41.   if m > n or m < 1 or n < 1: # Check if the user has entered a valid range
42.         print("Invalid input. Please enter valid numbers.")
43.     else:
44.         start_time = time.time() # Record the start time
45.         special_numbers = find_special_numbers(m, n) # Find special numbers
    within the range
46.         display_special_numbers(special_numbers) # Display the special numbers
47.          end_time = time.time() # Record the end time
48.  # Print the execution time
49.         print("Execution time: {:.6f} seconds".format(end_time - start_time))
50.
51.
52. if __name__ == "__main__":
53.     main()
54.
```

Student 1: Bryan's Solution - Explanation

Understanding The Problem:

In approaching this coding challenge, my initial step was to thoroughly understand the problem's requirements by carefully reading through the task description. The problem required a program capable of identifying a specific subset of numbers that are both prime—a fundamental concept in both mathematics and computer science—and palindromic, an interesting property where a number reads the same forwards as backwards. This intersection of number theory and string manipulation presents an intriguing computational task, as it necessitates the blend of numerical algorithms with string operations.

Approach to Solving the Problem,description and Highlights of Difference in Code:

My strategy was direct and purposeful: to implement two pivotal functions, `is_prime` and `is_palindrome`. The `is_prime` function embodies efficiency, using the mathematical principle that any non-prime number has a divisor no larger than its square root. This optimizes the primality check, minimizing iterations, a strategy often reserved for algorithmically complex problems. The `is_palindrome` function takes full advantage of Python's string manipulation prowess, succinctly checking if a number reads the same backward as forward. These functions work in tandem, forming a robust system to sieve out the special numbers we seek.

To structure my code with the same clarity and methodical approach that I learned from AI, particularly from engaging with ChatGPT, I wrapped the functionality within a main function. This central function serves as a command center, invoking each specific check in sequence and collating the qualifying numbers. This design philosophy mirrors the modularity and sequence found in AI's approach to problem-solving, where tasks are methodically processed through designated functions or 'modules.' It's this AI-

influenced approach to code structure that allowed me to ensure that the script not only met the functional requirements but did so with a design that's clear and maintainable.

Integrating the user interaction within this framework, I developed a `main` function that guides users through the input process with validation steps, an important practice that ensures the program's robustness against invalid input. Furthermore, the `display_special_numbers` function I conceptualized was designed to present the results to the user thoughtfully, adhering to the output specifications provided in the coursework brief. This holistic approach to solving the problem is a testament to my growth as a computer science student, demonstrating my ability to apply theoretical knowledge to develop practical and user-focused software solutions.

```python
def display_special_numbers(special_numbers):
    total = len(special_numbers)
    if total < 6:
        print("Total: Special Numbers =", total)
        print("List of special numbers =", special_numbers)
    else:
        print("Total: special Numbers =", total)
        print("List of special numbers =", special_numbers[:3] + special_numbers[-3:])
```

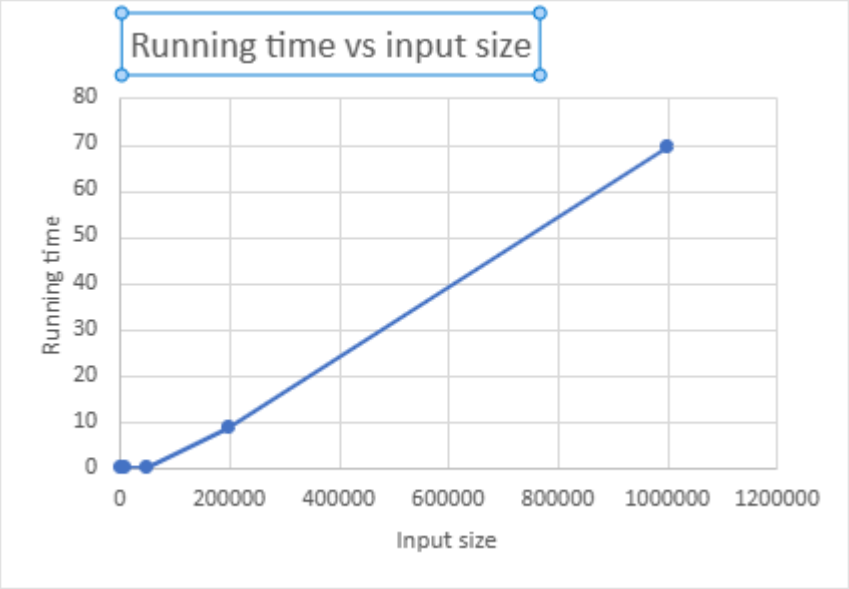
```python
def main():
    m = int(input("Enter the lower limit (m): "))
    n = int(input("Enter the upper limit (n): "))
    if m > n or m < 1 or n < 1:
        print("Invalid input. Please enter valid numbers.")
    else:
        start_time = time.time()
        special_numbers = find_special_numbers(m, n)
        display_special_numbers(special_numbers)
        end_time = time.time()
        print("Execution time: {:.6f} seconds".format(end_time - start_time))


if __name__ == "__main__":
    main()
```

Student 1: Bryan's Results:

#	Input(m)	Input(n)	Output	Running time (s)
1	1	2_000	Total: special Numbers = 20 List of special numbers = [2, 3, 5, 797, 919, 929]	0.001978 seconds

			Execution time: 0.001978 seconds	
2	100	10_000	Total: special Numbers = 15 List of special numbers = [101, 131, 151, 797, 919, 929] Execution time: 0.009050 seconds	0.009050 seconds
3	20_000	80_000	Total: special Numbers = 48 List of special numbers = [30103, 30203, 30403, 79397, 79697, 79997] Execution time: 0.083562 seconds	0.083562 seconds
4	100_000	2_000_000	Total: special Numbers = 190 List of special numbers = [1003001, 1008001, 1022201, 1993991, 1995991, 1998991] Execution time: 8.887144 seconds	8.887144 seconds
5	2_000_000	9_000_000	Total: special Numbers = 327 List of special numbers = [3001003, 3002003, 3007003, 7985897, 7987897, 7996997] Execution time: 69.357623 seconds	69.357623 seconds
6	10_000_000	100_000_000	Total: Special Numbers = 0 List of special numbers = [] Execution time: 2337.687889 seconds	2337.687889 seconds
7	100_000_000	400_000_000	N/A	N/A
8	1_100_000_000	15_000_000_000	N/A	N/A
9	15_000_000_000	100_000_000_000	N/A	N/A
10	1	1_000_000_000_000	N/A	N/A



Student 2: Nikmal's Solution:

Understanding The Problem:

I understood the problem by reading over the specification and the question a couple of times first to understand what the question is and what I must do. From reading over the specification first, I knew that we are in a group as part of a summer internship joining a growing start – up company in the cryptography domain. I knew that my group had been asked to solve a coding interview question in python that tests a person's knowledge in data structure and algorithms with different levels of optimisation. From this, I knew what was going on and that I had to solve a coding interview question in python. After I read the question a couple of times, I knew that the problem was that I had to create a python program where a user is asked to input two numbers m and n, where m is supposed to be smaller than n and then the program should print the special numbers between them and the amount of special numbers that there are. I knew that if there are 6 special numbers it should display all of them but if there are more than 6, (e.g. 10 special numbers) then it should show you the first three smallest and the last three largest special numbers. And I also found out that special numbers are supposed to be both prime and palindromic and anything else does not count as special numbers. From this I knew that I needed to have two functions to check for prime numbers and palindromic numbers.

Approach to Solving the Problem:

My approach to solving the problem was to create separate functions for each requirement that was needed. For example, I thought of having a function for finding special numbers which were supposed to be both prime and palindromic and from that I knew that I needed a function for checking for prime numbers and checking for palindrome numbers. I also thought that I should have had a function for displaying the special numbers and the last function was not really necessary, but I wanted to create a main function that should tell the user to input the two positive integers, which were m and n and also I thought of adding some requirements where some inputs where not allowed to be entered and there should

be an error if the user inputs these numbers. The requirements were in the question, for example, a user shouldn't enter a number where the number in m is larger than the number they put in n. After doing all of this, I knew that I should create a total of five functions maximum because I couldn't think of anything else for the question.

Description and Highlights of Difference in Code:

My code has a total of 5 functions. These functions were a function to find the special numbers (m and n), a function to check for prime numbers, a function to check for palindromes, a function that displays the special numbers, and a main function that runs everything and asks the user to input the two positive numbers m and n. In the prime function, I used a simple way to check for prime numbers and used the import math library. The prime function works by the number 2 being assigned to the variable a, and uses a while loop that continues as long as the variable a is less than or equal to the square root of the number that the user inputs. The loop checks if the number the user inputs is divisible by a, then it returns False and means that it is not a prime number and if it is not divisible by a then it returns True and means that it is a prime number. The palindrome function is simple and checks for palindrome numbers by changing the number n to a string and returning True or False is the string n is equal to the reverse of that string. The finding special number's function works by returning the list of special numbers if the values of m and n are both prime and palindrome numbers otherwise it doesn't. The display special numbers function just prints all the special numbers if they are less than 6 otherwise it will display the first three smallest and the last three largest special numbers. Finally, the main function just asks the user to input m and n and has some rules where the user can't enter a number where m is smaller than n and any negative numbers, then it just runs the find special numbers and displays the special numbers and times it in seconds. The difference in this code is that it has a total of 5 functions and also a main function, and the way that it finds the prime number is different because it uses the import math library and a different method from the others. It also has a different structure than the others. The main function then runs at the end. The full code can be viewed in the appendix below. Here is a screenshot of the difference in my solution compared to everyone else's:

```
1 usage
def is_prime(n):
    a = 2
    while a <= math.sqrt(n):
        if n % a < 1:
            return False
        a = a + 1
    return n > 1
```

```
1 usage
def main():
    m = int(input("Please enter a positive number (m): "))
    n = int(input("Please enter a bigger positive number (n): "))
    if m < 1 or n < 1:
        print("You can't enter a negative number")
    if m > n:
        print("M must be smaller than N")
    start = t.default_timer()
    special_numbers = find_special_numbers(m, n)
    print("Total number of special numbers:", len(special_numbers))
    display_special_numbers(special_numbers)
    end = t.default_timer()
    print(f"Run Time: {end - start} seconds")
```

Nikmal's Results:

Test Cases	Input(m)	Input(n)	Output	Running time (s)
1	1	2_000	Total: special Numbers = 20 First three smallest special numbers = [2, 3, 5] Last three largest special numbers = [797, 919, 929]	0.00059840000000332 seconds
2	100	10_000	Total: special Numbers = 15	0.00226780000000264 seconds

			First three smallest special numbers = [101, 131, 151] Last three largest special numbers = [797, 919, 929]	
3	20_000	80_000	Total: special Numbers = 48 First three smallest special numbers = [30103, 30203, 30403] Last three largest special numbers = [79397, 79697, 79997]	0.01463340000000013 seconds
4	100_000	2_000_000	Total: special Numbers = 190 First three smallest special numbers = [1003001, 1008001, 1022201] Last three largest special numbers = [1993991, 1995991, 1998991]	0.4176749000000006 seconds
5	2_000_000	9_000_000	Total: special	1.5889673000000002 seconds

			Numbers = 327 Firs three smallest special numbers = [3001003, 3002003, 3007003] Last three largest special numbers = [7985897, 7987897, 7996997]	
6	10_000_000	100_000_000	Total: Special Numbers = 0 All special numbers = []	18.873718 seconds
7	100_000_000	400_000_000	Total: special Numbers = 2704 Firs three smallest special numbers = [100030001, 100050001, 100060001] Last three largest special numbers = [399737993, 399767993, 399878993]	68.7384736 seconds
8	1_100_000_000	15_000_000_000	N/A	N/A
9	15_000_000_000	100_000_000_000	N/A	N/A
10	1	1_000_000_000_000	N/A	N/A

Student 3 – Louis Anthony's Solution:

```
55. import time # to import time so it can measure time
56.
57. class special_number: # Defines class named special_number
58.     def __init__(self, Prime, Palindrome):
59.         self.prime_num = Prime
60.         self.palindrome = Palindrome
61.
62.     def is_Prime(self, x): # Checks if a number is prime
63.         if x < 2:
64.             return False
65.         if x == 2:
66.             return True
67.         if x % 2 == 0:
68.             return False
69.         for i in range(3, int(x ** 0.5) + 1, 2):
70.                 return False
71.         return True
72.
73.     def is_Palindromic(self, x): # Check if a number is palindromic
74.         num = str(x)
75.         return num == num[::-1]
76.
77. # user inputs numbers
78. m = int(input('Enter lower range number:')) # Gets a low range input
79. n = int(input('Enter higher range number:')) # Gets a high range input
80.
81. start_time = time.time() # Records start time of the performance
82. special_numbers = [] # empty set
83.
84.
85. # used Chatgpt to help for this code
86. while m <= n: # while loop from lower to higher
87.     if special_number(m, m).is_Prime(m) and special_number(m,
    m).is_Palindromic(m): # checks from inputted m value if number is both prime
    and palindromic
88.         special_numbers.append(m) # if the value satisfies both, then it is
    added to the empty list
89.     m += 1 # moves onto the next value within the range
90.
91.
92.  print('There are', len(special_numbers), 'numbers')
93.
94. if len(special_numbers) > 6: # if statement on the amount of special numbers
95.     print('The first 3 special numbers are:', special_numbers[:3]) # this
    string :3 finds the first three in the empty list
96.     print('The last 3 special numbers are:', special_numbers[-3:]) # this
    string -3: finds the last three in the empty list
97. else:
98.     print('The special numbers are', special_numbers)
99.
100.       end_time = time.time() # Records end time of the performance
101.       time_taken = end_time - start_time
102.       print('Time taken:', time_taken, 'secs') # Prints time taken for
    execution
103.
```

Explanation

Understanding of the problem:

The problem given to us was to create a programme that helps find special numbers, which are numbers that are both prime numbers and palindromes.

From this given issue, I understood that the programme should be able to identify and classify what special numbers are. This means that I would have to add a code that identifies prime numbers, which are numbers that can only be divide by itself and 1 excluding 1 and 0, and palindrome numbers, numbers that are the same when they are in reverse order like 1001. Additionally, I would need to have a user interface for the range of numbers since it would need the user to input numbers for the programme to find these special numbers between the range of these inputted numbers. Along with these other understandings, I would have to put a limit to how many special numbers are outputted and displayed with the max being 6 special numbers, as well as having the programme run under an hour with it also printing the time it took for the code to be processed. From all this information, I understood what the assignment was and where to begin for making this programme.

Approaching the issue

With this understanding, it gave me an idea for how I should approach this situation. So, to begin with, I thought of using a class for special numbers to have them classified under one code so that when I try to produce the output, I can just put **special_number**. However, I would need to add a programme for identifying prime numbers and palindromes. Hence, I would add the attributes **Prime**, and **Palindrome** in the main section of the class. With this, I would have to make two sub-features for this class being **def is_Prime**, and **def is_Palindromic** to identify and define what are prime numbers and palindromic numbers. After this, I would need to make an empty list so that there is no limit to the numbers that can be in the list as well as a condition for any numbers to enter the empty list; I would name the list **special_numbers**. To finish the code, I would make a string of print statements that will output the number of special numbers as well as the numbers that satisfy the requirements to be one. Additionally, I would need to add timing into the code so that it can record the start and end time it took for the code to run since it needs to run under an hour.

Difference in this code compared to the others

The difference this code has from the others is the use of a class. I decided to add a class as to help classify special numbers so when it came to printing the output, the classification of a special number is already defined and considers the criteria needed for it to be a special number. Additionally, another difference in the code because of the class is the requirement needed to be subjected to being a special number if the first value is least than or equal to the second value.

Results

#	Input (m)	Input (n)	Output	Running time (s)
1	1	2_000	There are 20 numbers The first 3 special numbers	0.0029973983764648438 secs

			are: [2, 3, 5] The last 3 special numbers are: [797, 919, 929]	
2	100	10_000	There are 15 numbers The first 3 special numbers are: [101, 131, 151] The last 3 special numbers are: [797, 919, 929]	0.00751042366027832 secs
3	20_000	80_000	There are 48 numbers The first 3 special numbers are: [30103, 30203, 30403] The last 3 special numbers are: [79397, 79697, 79997]	0.05754232406616211 secs
4	100_000	2_000_000	There are 190 numbers The first 3 special numbers are: [1003001, 1008001, 1022201] The last 3 special numbers are: [1993991, 1995991, 1998991]	3.9644336700439453 secs
5	2_000_000	9_000_000	There are 327 numbers	26.91209363937378 secs

			The first 3 special numbers are: [3001003, 3002003, 3007003] The last 3 special numbers are: [7985897, 7987897, 7996997]	
6	10_000_000	100_000_000	There are 0 numbers The special numbers are []	901.0988442897797 secs
7	100_000_000	400_000_000	There are 2704 numbers The first 3 special numbers are: [100030001, 100050001, 100060001] The last 3 special numbers are: [399737993, 399767993, 399878993]	4695.734637022018 secs
8	1_100_000_000	15_000_000_000	N/A	N/A
9	15_000_000_000	100_000_000_000	N/A	N/A
10	1	1_000_000_000_000	N/A	N/A

Student 4 – Abdalla's Solution:

```
104.
105.
106.
107.
108.
109.
110.
111.
112.
113.
114.
115.
```

Explanation:

Understanding the problem

We were tasked with creating a programme that finds special numbers in between the users two inputs "M" and "N", for context the special numbers are numbers that are both prime numbers and palindromic numbers. Having to find the Intersection between the two required a thorough understanding of each individual number type.

Approaching the task

Realising the complexity of the task at hand and familiarising myself with both types of numbers I used various methods such as "SieveOfEratosthenes" which is a method for finding all prime numbers. Palindromic numbers posed a much more complicated issue and required a different approach needing the use of the function "is_palindrome" which checks if a string is a palindrome. With both working together it helped narrow down the search space for special numbers as shown in:

```
special_numbers = [num for num in range(m, n + 1) if is_prime(num, primes) and is_palindrome(num)]
```

Simply by defining what a prime number is and what a palindromic number is separately the process of finding the intersection between the two is made significantly easier. All dependant on the user inputs the programme will work to find the special numbers between

M and N, as a failsafe i added user input error checks for example if the user inputted the smaller number as the bigger number or vice versa it would print a reminder that it should be the other way around.

Difference in code

What i had done differently was at the end, the error checking allows a much more smoother user experience when using the programme. Excluding actual attempts at inputting wrong inputs, if the user had accidentally inputted something incorrect instead of the code breaking or ending it will prompt the user that the inputs are incorrect and they should fix that.

Abdalla **Results**

#	Input (m)	Input (n)	Output	Running time (s)
1	1	2_000	First 3 smallest special numbers: [1, 2, 3] Last 3 biggest special numbers: [1661, 1771, 1991] Total number of special numbers: 40	0.000999 seconds
2	100	10_000	First 3 smallest special numbers: [101, 121, 131] Last 3 biggest special numbers: [9449, 9559, 9889] Total number of special numbers: 42	0.005994 seconds

3	20_000	80_000	First 3 smallest special numbers: [30103, 30203, 30403] Last 3 biggest special numbers: [79697, 79897, 79997] Total number of special numbers: 101	0.032965 seconds
4	100_000	2_000_000	First 3 smallest special numbers: [1000001, 1003001, 1005001] Last 3 biggest special numbers: [1995991, 1996991, 1998991] Total number of special numbers: 415	1.096870 seconds
5	2_000_000	9_000_000	First 3 smallest special numbers: [3001003, 3002003, 3004003] Last 3 biggest special numbers: [7996997, 7998997, 7999997] Total number of special numbers: 724	4.652204 seconds

6	10_000_000	100_000_000	All 0 special numbers: [] Total number of special numbers: 0	69.219634 seconds
7	100_000_000	400_000_000	First 3 smallest special numbers: [100030001, 100050001, 100060001] Last 3 biggest special numbers: [399878993, 399949993, 399959993] Total number of special numbers: 5696	253.649488 seconds
8	1_100_000_000	15_000_000_000	N/A	N/A
9	15_000_000_000	100_000_000_000	N/A	N/A
10	1	1_000_000_000_000	N/A	N/A

Student 5 – Alana taiana's Solution:

```
import time
def is_prime(n):
    if n < 2:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    sqrt_n = int(n ** 0.5) + 1
    for i in range(3, sqrt_n, 2):
        if n % i == 0:
            return False
    return True


def is_palindrome(n):
    return str(n) == str(n)[::-1]


def find_special_numbers(m, n):
```

```
    special_numbers = [num for num in range(m, n + 1) if is_prime(num) and
is_palindrome(num)]
    return special_numbers


def display_special_numbers(special_numbers, m, n):
    if len(special_numbers) < 6:
        print("All special numbers between {} and {} are: {}".format(m, n,
special_numbers))
    else:
        print("First 3 smallest special numbers between {} and {} are: {}".format(m,
n, special_numbers[:3]))
        print("Last 3 biggest special numbers between {} and {} are: {}".format(m, n,
special_numbers[-3:]))
    print("Total number of special numbers between {} and {} is: {}".format(m, n,
len(special_numbers)))


def main():
    try:
        m = int(input("Enter the lower limit (m): "))
        n = int(input("Enter the upper limit (n): "))
        if m < 0 or n < 0 or m > n:
            print("Invalid input. Please enter positive numbers with m < n.")
            return
    except ValueError:
        print("Invalid input. Please enter valid integers.")
        return

    start_time = time.time()
    special_numbers = find_special_numbers(m, n)
    end_time = time.time()

    display_special_numbers(special_numbers, m, n)
    print("Elapsed time:", end_time - start_time, "seconds")


if __name__ == "__main__":
    main()
```

Understanding of the problem:

Based on the requirement given, what I understood is that we need a python program that allows users to input two positive numbers *m* and *n* (where *m* is smaller than *n*) and finds and displays the special numbers between them. Special numbers are defined as numbers that are both prime and palindromic. The program should not use a hard-coded list of values or an external data file, it must aim to finish with in 1 hour of runtime for any test case and the solution should be concise, not exceeding 100 lines. The program code language that will be used is python.

Approach to solve the problem:

To answer the problem, we must locate and display numbers between the supplied positive integers m and n. And there are special numbers that are both prime and palindromic. The first technique I used was to implement the *'is_prime'* function, which checks if an integer is prime. I utilized the trial division approach to quickly determine whether an integer is divisible by any number other than one and itself. This function returns true when the number is prime and false otherwise. Next, I created a function *'is_palindrome'* to determine whether a number is palindromic. I turned the number to a string and compared it to the reverse. If they are equivalent, then the number is a palindrome. After, I created a function called *'find_special_numbers'* that iterates overall numbers in the range m to n (inclusive). It uses the helper functions to determine whether a given number is both prime and palindromic and if this is the case, it is added to a special number list. In the *'main'* function, I prompt the user to enter m and n. Then, I use the *'find_special_numbers'* function to obtain a list of special numbers. Depending on the quantity of special numbers, I either print them all or print the first three smallest and last three largest special numbers. To measure the running time of the solution for each test case, I used the *'time'* module to record the start and end times of the execution and calculated the elapsed time.

Difference in my code:

The difference in my code is the display special numbers, where the outputs (first 3 smallest numbers, last 3 biggest and total number of special numbers) are printed with the m and n respectively.

```
Enter the lower limit (m): 100
Enter the upper limit (n): 120000
First 3 smallest special numbers between 100 and 120000 are: [101, 131, 151]
Last 3 biggest special numbers between 100 and 120000 are: [97879, 98389, 98689]
Total number of special numbers between 100 and 120000 is: 108
```

Results

#	Input(m)	Input(n)	Output	Running time (s)
1	1	2000		
2	100	10_000	First 3 smallest special numbers between 100 and 10000 are: [101, 131, 151]	Elapsed time: 0.008484840393066406 seconds

			Last 3 biggest special numbers between 100 and 10000 are: [797, 919, 929] Total number of special numbers between 100 and 10000 is: 15	
3	20_000	80_000	First 3 smallest special numbers between 20000 and 80000 are: [30103, 30203, 30403] Last 3 biggest special numbers between 20000 and 80000 are: [79397, 79697, 79997] Total number of special numbers between 20000 and 80000 is: 48	Elapsed time: 0.050141096115112305 seconds
4	100_000	2_000_000	First 3 smallest special numbers between 100000 and 2000000 are: [1003001, 1008001, 1022201] Last 3 biggest special numbers between	Elapsed time: 5.383028268814087 seconds

			100000 and 2000000 are: [1993991, 1995991, 1998991] Total number of special numbers between 100000 and 2000000 is: 190	
5	2_000_000	9_000_000	First 3 smallest special numbers between 2000000 and 9000000 are: [3001003, 3002003, 3007003] Last 3 biggest special numbers between 2000000 and 9000000 are: [7985897, 7987897, 7996997] Total number of special numbers between 2000000 and 9000000 is: 327	Elapsed time: 44.02989387512207 seconds
6	10_000_000	100_000_000	All special numbers between 10000000 and 100000000 are: [] Total number of special numbers between	Elapsed time: 1701.9319767951965 seconds

			10000000 and 100000000 is: 0	
7	1_100_000_000	400_000_000	N/A	N/A
8	1_100_000	15_000_000_000	N/A	N/A
9	15_000_000_000	100_000_000_000	N/A	N/A
10	1	1_000_000_000_000	N/A	N/A

1.1 compare 2 students codes:

Student 3 – Louis Anthony's_ Code

```
import time  # Import the time module for measuring execution time

class special_number:  # Define a class named special_number
    def __init__(self, Prime, Palindrome):
        self.prime_num = Prime
        self.palindrome = Palindrome

    def is_Prime(self, x):  # Method to check if a number is prime
        if x < 2:
            return False
```

Student 1 - Bryan's___ **Code**

```
import time


def is_prime(n):

    if n < 2:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True
```

```
        if x == 2:
            return True
        if x % 2 == 0:
            return False
        for i in range(3, int(x ** 0.5) + 1,
2):
            if x % i == 0:
                return False
        return True

    def is_Palindromic(self, x):  # Method
to check if a number is palindromic
        num = str(x)
        return num == num[::-1]

m = int(input('Enter lower range number:'))
# Get lower range input from the user
n = int(input('Enter higher range number:'))
# Get higher range input from the user

start_time = time.time()  # Record the start
time for performance measurement
special_numbers = [] # empty set

# used Chatgpt to help for this code
while m <= n:
    if special_number(m, m).is_Prime(m) and
special_number(m, m).is_Palindromic(m):
        special_numbers.append(m)
    m += 1

print('There are', len(special_numbers),
'numbers')

if len(special_numbers) > 6:  # if statement
on the amount of special numbers
    print('The first 3 special numbers
are:', special_numbers[:3]) # this string :3
finds the first three in the empty list
    print('The last 3 special numbers are:',
special_numbers[-3:]) # this string -3:
finds the last three in the empty list
else:
    print('The special numbers are',
special_numbers)

end_time = time.time()  # Record the end
time for performance measurement
time_taken = end_time - start_time
print('Time taken:', time_taken, 'secs')  #
Print the time taken for execution
```

```
def is_palindrome(n):

    return str(n) == str(n)[::-1]


def find_special_numbers(m, n):

    special_numbers = []
    for num in range(m, n + 1):
        if is_prime(num) and
is_palindrome(num):
            special_numbers.append(num)
    return special_numbers


def
display_special_numbers(special_numbers
):
    total = len(special_numbers)
    if total < 6:
        print("Total: Special Numbers
=", total)
        print("List of special numbers
=", special_numbers)
    else:
        print("Total: special Numbers
=", total)
        print("List of special numbers
=", special_numbers[:3] +
special_numbers[-3:])


def main():
    m = int(input("Enter the lower
limit (m): "))
    n = int(input("Enter the upper
limit (n): "))
    if m > n or m < 1 or n < 1:
        print("Invalid input. Please
enter valid numbers.")
    else:
        start_time = time.time()
        special_numbers =
find_special_numbers(m, n)

display_special_numbers(special_numbers
)
        end_time = time.time()
        print("Execution time: {:.6f}
seconds".format(end_time - start_time))


if __name__ == "__main__":
    main()
```

Comparison:

Similarities-

From these 2 codes, the similarities are as follows:

- **Purpose:** Both codes aim to find numbers within a specified range (between `m` and `n`) that are both prime and palindromic. They achieve this by defining and utilizing functions (or methods in the case of Louis Anthony's code) to check for prime numbers and palindromic numbers.
- **Input from the User:** Each script requests input from the user for the lower and upper limits (`m` and `n`) to define the range within which the special numbers (prime and palindromic) are sought.
- **Prime Number Checking Logic:** The logic for determining whether a number is prime is conceptually similar in both scripts. They both exclude numbers less than 2, identify 2 as prime, and then check for divisibility by numbers up to the square root of the target number.
- **Palindromic Number Checking Logic:** Both scripts use a similar strategy to check if a number is palindromic by converting the number to a string and comparing it to its reverse.
- **Output:** Both codes eventually print the found special numbers and provide some form of timing for the execution of the script, highlighting the performance aspect.

Differences-

Besides these similarities, there are some differences, they are a follow:

- **Approach and Structure:** Louis Anthony's code uses a class-based approach, defining a `special_number` class with methods to check for prime and palindromic numbers. In contrast, Bryan's code uses a functional approach, with separate functions defined for checking prime numbers, checking palindromic numbers, finding the special numbers, displaying them, and wrapping the main logic in a `main` function.
- **Performance Measurement:** Louis Anthony's code directly calculates and prints the execution time at the end of the script, using a straightforward subtraction of start and end times measured with `time.time()`. Bryan's code also measures execution time using `time.time()`, but it formats the output more neatly with a formatted string to show execution time up to six decimal places.
- **Handling of Special Cases and Errors:** Bryan's code includes a specific check to ensure that the user inputs are valid (i.e., `m` and `n` are positive and `m` is less than or equal to `n`). This adds robustness by preventing the script from proceeding with invalid inputs. Louis Anthony's code does not include explicit input validation.
- **Object-Oriented vs. Procedural:** The use of a class in Louis Anthony's code is a clear distinction, leaning towards an object-oriented programming (OOP) style. This encapsulates the prime and palindromic checking functionality within a class structure. Bryan's procedural approach is more straightforward and functional, without encapsulating the functionality within a class.
- **Code Optimization and Efficiency:** While both scripts are designed to achieve similar outcomes, the object-oriented approach in Louis Anthony's code might be less efficient

due to the repeated instantiation of the **`special_number`** class for each number in the range. Bryan's functional approach directly calls the functions without the overhead of object creation, which could potentially offer better performance for large ranges.

- **Output Formatting and Presentation:** The way the special numbers are presented to the user differs slightly. Louis Anthony's code prints the special numbers directly and adds an if-else statement to differentiate the output based on the count of special numbers found. Bryan's code also includes this differentiation but combines the first three and last three special numbers into a single list for presentation if more than six special numbers are found, providing a succinct overview of the results.

2. Test and analyse your solution!

Your test cases:

Nikmal's Test Cases:

Test cases	Input (m)	Input (n)	Output	Justification	Nikmal's results in seconds
1	5_000_000	50_000_000	Total number of spcial numbers = 306 First three smallest special numbers = [7014107, 7035307, 7036307] Last three largest special numbers = [9980899, 9981899, 9989899]	The reason that I have chosen these inputs for the first test case is that my code was already quite fast and running until tesst case 7 so I wanted to start off with a larger input and keep increasing it on each test case.	9.4020478 seconds
2	10_000_000	100_000_000	Total number of spcial numbers = 0 All pecial numbers = []	These inputs were chosen for test case 2 is because it demonstrates the correctness of my solution, and I was	18.3875575 seconds

				increasing it more and more since my code ran the first test case in 9 seconds which was quite fast.	
3	25_000_000	150_000_000	Total number of spcial numbers = 736 First three smallest special numbers = [100030001, 100050001, 100060001] Last three largest special numbers = [149909941, 149919941, 149939941]	These test cases were chosen for test case 3 because test case 2 ran double the amount it did for test case 1, but because the inputs were way larger for test case 2, it still managed to run it in 18 seconds, so I decided to increase it more.	27.6627808 seconds
4	150_000_000	450_000_000	Total number of spcial numbers = 1968 First three smallest special numbers = [150070051, 150151051, 150181051] Last three largest special numbers = [399737993, 399767993, 399878993]	These inputs were chosen for test case 4 because the third test case was still fast and it ran in 27 seconds, therefore this time I decided to increase the input way higher.	68.4293212 seconds
5	200_000_000	600_000_000	Total number of spcial numbers = 1280	Finally, this was the last test	87.8879261 seconds

			First three smallest special numbers = [300020003, 300080003, 300101003] Last three largest special numbers = [399737993, 399767993, 399878993]	case that I decided to create, and this test case was chosen because the code was still able to handle all the other test cases, therefore I wanted to check how high the highest input goes before it takes a long time to run.	

Louis Anthony's Test Cases:

Test cases	Input (m)	Input (n)	Output	Justification	Louis Anthony's results in seconds
1	990000001	990000004	There are 0 numbers The special numbers are [] Time taken: 0.0 secs	I chose these numbers as I wanted to identify if the difference in the unit's matter. Which is yes as although the value is high nearing the billions, it still ran quickly	0.0 secs

				showing every unit matters.	
2	13338	72622	There are 47 numbers The first 3 special numbers are: [13831, 13931, 14341] The last 3 special numbers are: [71317, 71917, 72227] Time taken: 0.05640053749084473 secs	Just like with the first test, I used random numbers which have random units to test the effectivity of the code. This has resulted in a fast time as the numbers are not as high along with the difference between them.	0.05640053749084473 secs
3	5_000_000	50_000_000	There are 306 numbers The first 3 special numbers are: [7014107, 7035307, 7036307] The last 3 special numbers are: [9980899, 9981899, 9989899] Time taken: 297.19819736480713 secs	The third test case, I wanted to see how long it would take my code to run one of my groupmates inputted range. Since the running time of my groupmates was quick so	297.19819736480713 secs

				I wanted to see if it would be quick for my code as well which it was not.	
4	25_000_000	150_000_000	There are 736 numbers The first 3 special numbers are: [100030001, 100050001, 100060001] The last 3 special numbers are: [149909941, 149919941, 149939941] Time taken: 1315.529902458191 secs	This test was used as I wanted to push the codes limits and whether it finishes within an hour. This is since the code took a while to produce an output, but it was able to give a run time that is within an hour but still long.	1315.529902458191 secs
5	200_000_000	600_000_000	There are 1280 numbers The first 3 special numbers are: [300020003, 300080003, 300101003] The last 3 special numbers are: [399737993, 399767993, 399878993] Time taken: 7846.754552364349 secs	For this final test case, I wanted to see if the code could handle a large number that has a large gap between	7846.754552364349 secs

				the to, However, it took a while for it to produce an output going over the time limit of an hour.	

Bryan's Test Cases:

Test cases	Input (m)	Input (n)	Output	Justification	Bryan's results in seconds
1	1_000	10_000	Total: Special Numbers = 0 List of special numbers = [] Execution time: 0.007069 seconds	The main reason i shouce a large input size is as my code was runuing fast before witha smallere thi time i wante to do a lrger one, until case test 5 to 6 it starte to take a long time to process the input	0.007069 seconds
2	2,000	200,000	Total: special Numbers = 93 List of special numbers = [10301, 10501, 10601, 97879, 98389, 98689] Execution time: 0.309811 seconds	These case inputs were choice as in test 1 run was fat, so the input was double to see the difference in speed	0.309811 seconds

3	5,000	500_000	Total: special Numbers = 93 List of special numbers = [10301, 10501, 10601, 97879, 98389, 98689] Execution time: 1.096935 seconds	This was choice as in test1 to 2 it was running speed was still in 0.0.... I wanted to increase it again as it was running cuit fast. Also to see if increasing it a little more will make it reach 1.0	1.096935 seconds
4	10,000	10_000_000	Total: special Numbers = 761 List of special numbers = [10301, 10501, 10601, 9980899, 9981899, 9989899] Execution time: 75.463877 seconds	The input was choice before in test 3 I was able to see my program reach1.0... seconds and I wanted to see if I double the input how long would the program take	75.463877 seconds
5	15_000	50_000_000	Total: special Numbers = 748 List of special numbers = [15451, 15551, 16061, 9980899, 9981899, 9989899] Execution time: 721.477651 seconds	Finally, this input was choice a the program was able to handle tyhe lat test o i wanted to ee up to what point will the program tart to take a long time to run it.	721.477651 seconds

Abdalla's Test cases

Test cases	Input (m)	Input (n)	Output	Justification	Abdalla's results in seconds
1	1	1_000_000	First 3 smallest special numbers: [1, 2, 3] Last 3 biggest special numbers: [99299, 99599, 99899] Total number of special numbers: 183	Large gap to prove the programmes efficacy	0.572407 seconds
2	1_000_000	2_000_000	First 3 smallest special numbers: [1000001, 1003001, 1005001] Last 3 biggest special numbers: [1995991, 1996991, 1998991] Total number of special numbers: 415	the same gap but used with biggers numbers to show that the code can still work with a large pool of numbers	0.554426 seconds
3	1	1000	First 3 smallest special numbers: [1, 2, 3] Last 3 biggest special numbers: [959, 979, 989] Total number of special numbers: 33	Essentially a test of how fast the code could work with typical small numbers	0.000999 seconds

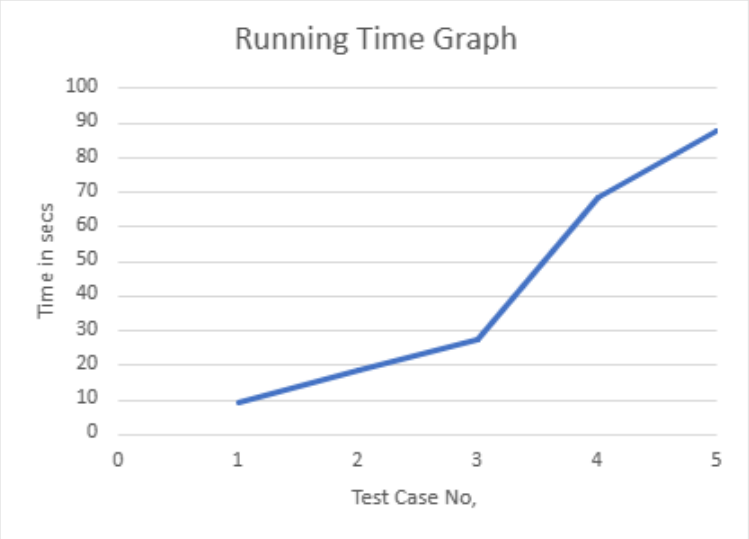
4	275663	1098674	First 3 smallest special numbers: [1000001, 1003001, 1005001] Last 3 biggest special numbers: [1085801, 1092901, 1093901] Total number of special numbers: 41	An unbiased input totally randomised to prove programmes authenticity	0.439546 seconds
5	10_000_000	1_000_000	Error: m should be smaller than n.	To test incorrect input response	N/A

Alana's test cases

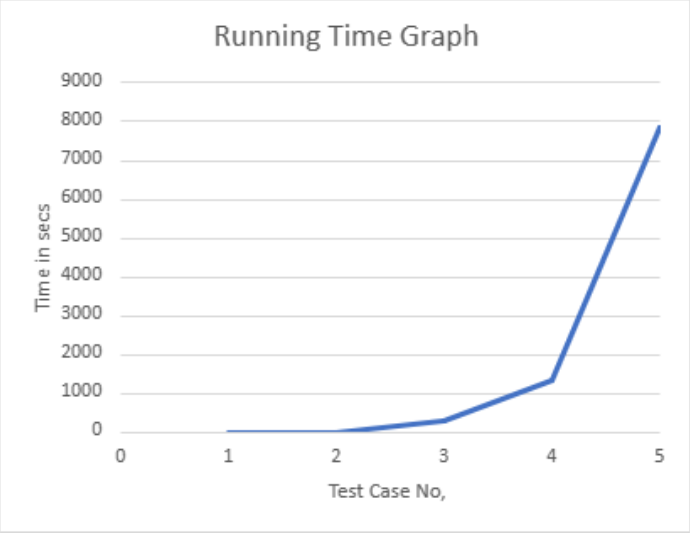
testcase	Input(m)	Input(n)	Output	Justification	Alana's results
1	1	1_000_000	First 3 smallest special numbers between 1 and 1000000 are: [2, 3, 5] Last 3 biggest special numbers between 1 and 1000000 are: [97879, 98389, 98689] Total number of special numbers between 1 and 1000000 is: 113	An extensive range of number to see how it performs with a considerable number of interactions	Elapsed time: 2.035788059234619 seconds
2	-20_000	500_000	Invalid input. Please enter positive numbers with m < n.	Test the program to ensure that m < n to ensure it handles invalid input gracefully and provides appropriate error massages	N/A

3	1	200	First 3 smallest special numbers between 1 and 200 are: [2, 3, 5] Last 3 biggest special numbers between 1 and 200 are: [151, 181, 191] Total number of special numbers between 1 and 200 is: 10	Small range of number to test how fast the code runs	Elapsed time: 0.0001592636108398 4375 seconds
4	12_000	100_000	First 3 smallest special numbers between 12000 and 100000 are: [12421, 12721, 12821] Last 3 biggest special numbers between 12000 and 100000 are: [97879, 98389, 98689] Total number of special numbers between 12000 and 100000 is: 88	2	Elapsed time: 0.0759980678558349 6 seconds
5	12_000	100_000	First 3 smallest special numbers between 12000 and 100000 are: [12421, 12721, 12821] Last 3 biggest special numbers between 12000 and 100000 are: [97879, 98389, 98689] Total number of special numbers between 12000 and 100000 is: 88	Execute the code multiple time with the same input to see if there are any variations in execution time due to external factors	Elapsed time: 0.0781922340393 0664 seconds

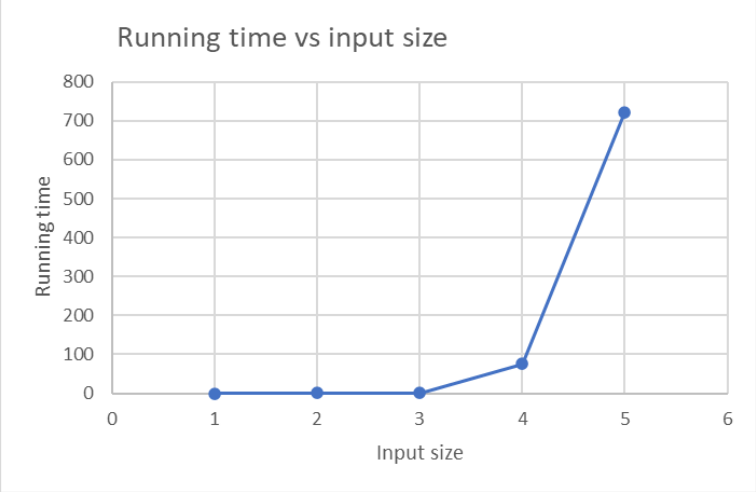
Running time graphs Nikmal's Graph:



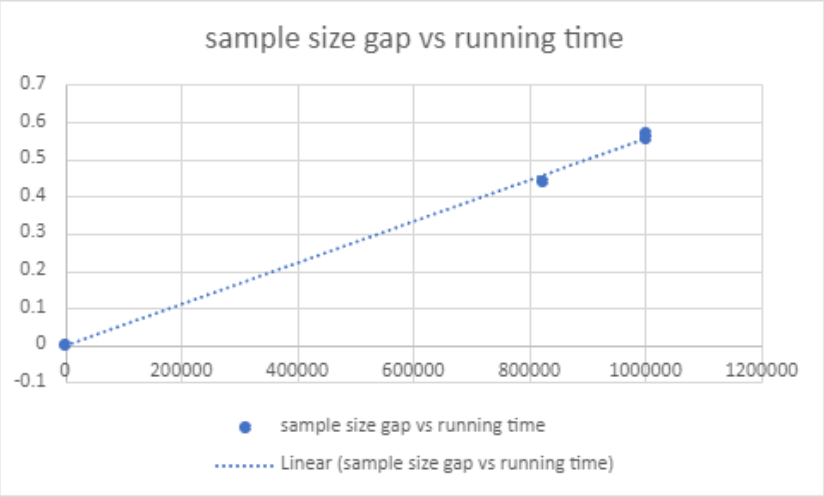
Louis Anthony's Running time graphs:



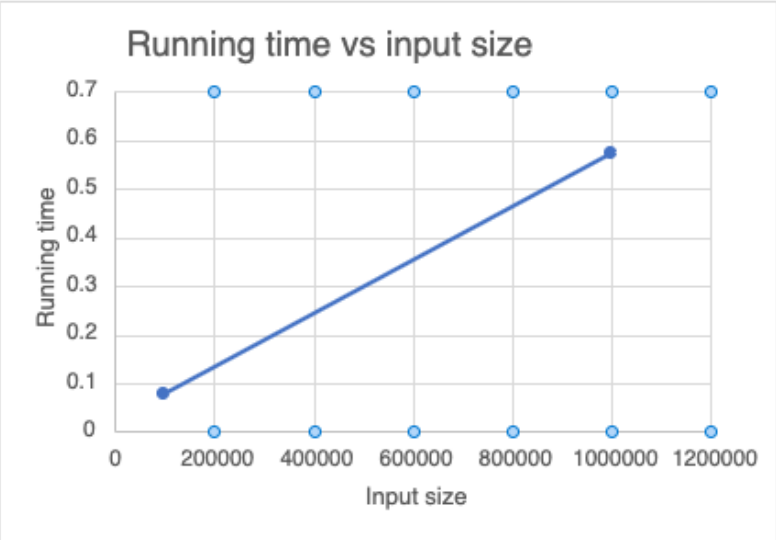
Bryan's Running time graphs:



Abdalla's Running time graphs:



Alana taiana's Running time graphs:



Complexity analysis

Students	Time Complexity	Big-O Notation
Nikmal	Linear increase with test cases	O(n)
Louis Anthony	Exponential increase at last test case	O(2^n) or O(n!)
Bryan	Sudden increase at larger input sizes	O(n^k), k > 1
Abdalla	Linear increase with input size	O(n)
Alana	Linear increase with input size	O(n)

- **Nikmal's graph**: The running time appears to increase linearly with the test case number. This suggests a time complexity that could be O(n).

- **Louis Anthony's graph**: This graph shows a dramatic increase in running time at test case 5, which suggests a possible exponential time complexity, potentially O(2^n) or O(n!).
- **Bryan's graph**: Shows a jump in running time at input size 5 and 6, which could be indicative of a polynomial time complexity, like O(n^k) for some k > 1.
- **Abdalla's graph**: Depicts a gradual increase in running time with the sample size gap. It appears to be a linear relationship, suggesting O(n).
- **Alana's graph**: Also appears to show a linear relationship, suggesting O(n).

3. Optimise solutions!

Solution 1-2:

Which ones did you group choose and give reasons for all the optimising steps that your group took.

Short description and highlights of the improvement in your code, and the full code in the Appendix.

Student 1 – Nikmal's Improved Code:

The first code that our group decided to improve on was Nikmal's code. The reason for this was because when everyone had finished coding their part and running the test cases from 1 to 10, out of all the members, Nikmal's code was running the fastest and was running until test case 7. However, the other group members code was running until test case 5 and much slower than Nikmal's. This therefore lead everyone to choose Nikmal's code to improve on since it had the fastest run time for each test case in the test case table. After everyone had chosen Nikmal's code to improve on, each group

member then worked on improving the code and finding different solutions and ways to make it better and faster.

Highlights of Improvement in Code:

```
import timeit as t
import math

def find_special_numbers(m, n):
  special_numbers = []
      for length in range(len(str(m)), len(str(n)) + 1):
        palindromes = generate_palindromes(length)
        for num in palindromes:
            if m <= num <= n and is_prime(num):
                special_numbers.append(num)
    return special_numbers


def is_palindromic(n):
    return str(n) == str(n)[::-1]

def generate_palindromes(length):
    if length == 1:
        return range(1, 10)
    palindromes = []
    if length % 2 == 0:
        half_length = length // 2
        for num in range(10 ** (half_length - 1), 10 ** half_length):
            s = str(num)
            palindromes.append(int(s + s[::-1]))
    else:
        half_length = (length - 1) // 2
        for num in range(10 ** (half_length - 1), 10 ** half_length):
            s = str(num)
            for mid_digit in range(10):
                palindromes.append(int(s + str(mid_digit) + s[::-1]))
    return palindromes

def is_prime(n):
    if n <= 1:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    max_divisor = math.floor(math.sqrt(n))
    for d in range(3, max_divisor + 1, 2):
        if n % d == 0:
            return False
    return True
```

```
def display_special_numbers(special_numbers):
    if len(special_numbers) < 6:
        print("All special numbers:", special_numbers)
    else:
        print("First three smallest special numbers", special_numbers[:3])
        print("Last three largest special numbers", special_numbers[-3:])

def main():
    m = int(input("Please enter a positive number (m): "))
    n = int(input("Please enter a bigger positive number (n): "))
    if m < 1 or n < 1 or m > n:
        print("You can't enter a negative number and M must be smaller than N")
    start = t.default_timer()
    special_numbers = find_special_numbers(m, n)
    print("Total number of special numbers:", len(special_numbers))
    display_special_numbers(special_numbers)
    end = t.default_timer()
    print(f"Run time: {end - start} seconds")

if __name__ == "__main__":
    main()
```

Description of Improved Code

As you can see in the code above, the highlighted areas are the parts of the code that we changed to improve the code mainly and make it run faster. The code itself is similar to the main code in task 1 and not much was changed. However, we found out that there were two functions in the code that needed to be changed in order to make it run faster and all of the test cases in the table. The first function that we found was very slow was the is_prime function because we did a lot of research and found that the prime function that was being used was quite slow and not that efficient. The second function that we found was slowing the program down was the palindrome's function because it took time to change the input to a string and then check if it was equal to the reverse of that string. We knew that these two functions were not good with larger inputs. Our group then did research and tried to find different ways to change the prime and palindromes function to make it run faster and be able to handle larger inputs. From doing this, we managed to find a different way to check for prime numbers. This works by checking if n is less than or equal to 1, then it returns false because prime numbers are higher than 1. Then it checks if n is equal to 2 and that also returns True because 2 is a prime number. Then it checks if n is even except for 2 then it returns False because all even numbers higher than 2 are not prime. After it checks if n can be divided by any of the odd numbers starting from 3, up to the square root of n. Finally, after doing this it returns False if n is not a prime and returns True if n is a prime. The prime function was changed to this and when we ran the test cases, it still wasn't running all of them but we found that it was much faster than the other prime function but still not much difference. Then, we decided that in the find special numbers function, instead of checking if the number is prime then palindrome, we changed the order so it check if m and n are palindromes first then primes. This made the performance faster but we still couldn't get to the last 3 test cases.

After doing more research and asking friends for advice, we found that the palindrome function was slowing things down and that we should generate them instead of going number by number. After trying to find different ways to generate palindromes, Nikmal managed to find a palindrome function that generates them. This function works by checking if the length is 1, then it returns a range from 1 to 9. Then it checks if the length is even, it generates palindrome numbers by iterating over a range of numbers starting from '10 ** (half_length – 1) up to 10 ** half_length – 1, where half_length is the length divided by 2'. For every number in this range, it converts it to a string, then concatenates it with it's reverse, which is used to create palindrome numbers by combining strings representing numbers and their reverse forms. And then it converts the resulting string back to an integer, creating even-lengthed palindrome numbers. If the length is odd, it generates palindrome numbers by iterating over the same numbers as it does for even but where half_length is (length – 1) divided by 2. For ever number in this range, it converts it to a string, concatenates it with every possible digit and then it's revers, creating odd-lengthed palindrome numbers. The is_palindromic function was remained the same. After we had changed the palindrome function to generate them instead of going number by number, when we ran all of the test cases, the program was exceptionally faster and managed to run all of the test cases with the last test case being able to run in 4 and a half minutes. Then we realised that the old palindrome function was much slower and it was much faster to generate them instead. After doing all of this, we managed to successfully improve the code and be able to run all of the test cases in the table, with the last test case being able to run in about 5 minutes.

Nikmal's Results

Test cases	Input (m)	Input (n)	Output	Correctness	Running
1	1	2_000	Total number of special numbers: 20 First three smallest special numbers [2, 3, 5] Last three largest special numbers [797, 919, 929] Run time: 0.0001732000000000076163 seconds	Changed the prime number function to a different one and changed the palindrome function by adding another function to generate plaindromes to make it more faster.	0.000173200 sec
2	100	10_000	Total number of special numbers: 15 First three smallest special numbers [101, 131, 151]	Changed the prime number function to a different	0.00068070 sec

			Last three largest special numbers [797, 919, 929] Run time: 0.0006807000000002006 seconds	one and changed the palindrome function by adding another function to generate plaindromes to make it more faster.	
3	20_000	80_000	Total number of special numbers: 48 First three smallest special numbers [30103, 30203, 30403] Last three largest special numbers [79397, 79697, 79997] Run time: 0.0007754000000002037 seconds	Changed the prime number function to a different one and changed the palindrome function by adding another function to generate plaindromes to make it more faster.	0.00077540 sec
4	100_000	2_000_000	Total number of special numbers: 190 First three smallest special numbers [1003001, 1008001, 1022201] Last three largest special numbers [1993991, 1995991, 1998991] Run time: 0.00861150000000066 seconds	Changed the prime number function to a different one and changed the palindrome function by adding another function to generate plaindromes to make it more faster.	0.0086115 sec
5	2_000_000	9_000_000	Total number of special numbers: 327	Changed the prime number	0.0189358 sec

			First three smallest special numbers [3001003, 3002003, 3007003] Last three largest special numbers [7985897, 7987897, 7996997] Run time: 0.018935899999999783 seconds	function to a different one and changed the palindrome function by adding another function to generate plaindromes to make it more faster.	
6	10_000_000	100_000_000	Total number of special numbers: 0 All special numbers: [] Run time: 0.039189400000000596 seconds	Changed the prime number function to a different one and changed the palindrome function by adding another function to generate plaindromes to make it more faster.	0.03918940 sec
7	100_000_000	400_000_000	Total number of special numbers: 2704 First three smallest special numbers [100030001, 100050001, 100060001] Last three largest special numbers [399737993, 399767993, 399878993] Run time: 0.8021062000000008 seconds	Changed the prime number function to a different one and changed the palindrome function by adding another function to generate plaindromes to make it more faster.	0.802106 sec

8	1_100_000_000	15_000_000_000	Total number of special numbers: 5474 First three smallest special numbers [10000500001, 10000900001, 10001610001 Last three largest special numbers [14998289941, 14998589941, 14998689941 Run time: 16.787949400000002 seconds	Changed the prime number function to a different one and changed the palindrome function by adding another function to generate plaindromes to make it more faster.	16.787949 sec
9	15_000_000_000	100_000_000_000	Total number of special numbers: 36568 First three smallest special numbers [15001010051, 15002120051, 15002320051 Last three largest special numbers [99998189999, 99998989999, 99999199999 Run time: 223.3878821 seconds	Changed the prime number function to a different one and changed the palindrome function by adding another function to generate plaindromes to make it more faster.	223.38788
10	1	1_000_000_000_000	Total number of special numbers: 47995 First three smallest special numbers [2, 3, 5] Last three largest special numbers [99998189999, 99998989999, 99999199999 Run time: 254.1119021 seconds	Changed the prime number function to a different one and changed the palindrome function by adding another function to generate plaindromes to make it	254.11190

				more faster.	

Student 2 – Abdalla's Imrpoved Code:

The second code that our group decided to choose to improve on was Abdallah's code. The reason for this was because after running everyone's results and finding that Nikmal's code was the fastest one out of all of them, the second code that was the fastest was Abdallah's code. Abdallah's code was running until test case 5, which was the same test case that the others were running up to, except for Nikmal's. However, Abdallah's code was more faster in test case 5 than Bryan's, Louis's, and Alana's code. This then lead everyone to start working on improving Abdallah's code to get it to run all of the test cases in the table and then we would have a check-up on what everyone has done.

Highlights of Improvement in Code:

```
import time

def is_prime(num, primes):
   if num < 2:
     return False
    for prime in primes:
        if prime * prime > num:
            return True
        if num % prime == 0:
            return False
    return True

def sieve_of_eratosthenes_segmented(limit):
    sieve = [True] * (limit + 1)
    sieve[0] = sieve[1] = False

    for num in range(2, int(limit**0.5) + 1):
        if sieve[num]:
            for multiple in range(num*num, limit + 1, num):
                sieve[multiple] = False

    primes = [num for num in range(2, limit + 1) if sieve[num]]

    return primes

def is_palindrome(num):
    return str(num) == str(num)[::-1]

def generate_palindromes(length):
    if length == 1:
        return range(1, 10)
    palindromes = []
    if length % 2 == 0:
        half_length = length // 2
        for num in range(10 ** (half_length - 1), 10 ** half_length):
            s = str(num)
```

```
            palindromes.append(int(s + s[::-1]))
    else:
        half_length = (length - 1) // 2
        for num in range(10 ** (half_length - 1), 10 ** half_length):
            s = str(num)
            for mid_digit in range(10):
                palindromes.append(int(s + str(mid_digit) + s[::-1]))
    return palindromes

def find_special_numbers(m, n):
    start_time = time.time()

    sqrt_n = int(n ** 0.5) + 1
    primes = sieve_of_eratosthenes_segmented(sqrt_n)

    max_palindrome_length = len(str(n))
    palindromes = []
    for length in range(1, max_palindrome_length + 1):
        palindromes.extend(generate_palindromes(length))
    palindromes = [palindrome for palindrome in palindromes if m <= palindrome <=
n]

    special_numbers = [num for num in palindromes if is_prime(num, primes)]

    if len(special_numbers) <= 5:
        print(f"All {len(special_numbers)} special numbers: {special_numbers}")
    else:
        print(f"First 3 smallest special numbers: {special_numbers[:3]}")
        print(f"Last 3 biggest special numbers: {special_numbers[-3:]}")
    print(f"Total number of special numbers: {len(special_numbers)}")

    end_time = time.time()
    print(f"Run time: {end_time - start_time:.6f} seconds")

if __name__ == "__main__":
    m = int(input("Enter the smaller number (m): "))
    n = int(input("Enter the larger number (n): "))

    if m >= n:
        print("Error: m should be smaller than n.")
    else:
        find_special_numbers(m, n)
```

Description of Improved Code

As you can see in the code above, the highlighted parts of the code is what our group changed in the code to improve it and make it run faster. This code is also similar to Abdallah's main code and there wasn't a lot of changes that were made to make the code run faster. This is because the prime function that Abdallah had used was very fast for checking for prime numbers. He had used an algorithm which was the sieve_of_eratosthenes. This algorithm works by finding all prime numbers up to a given limit. It marks the multiple of each prime number starting from the number 2, since 2 is the lowest prime number. This function generates a list of prime numbers up to the square root of the given limit. This prime algorithm was actually what was making the code faster. So we had decided to leave the prime function as it is and not change it because the algorithm that

Abdallah was using was already quite fast. However, we noticed that the palindrome function was the same palindrome function that Nikmal used in his first code, which was changing the given number into a string and then comparing that string number with it's reverse to check for palindrome numbers. From this, we already knew that this is what made the code slower so we had decided to look for a different palindrome function to make the code run faster. After trying different palindrome functions and not being able to find a function that improved the code, we decided to use the same palindrome function that was used in Nikmal's improved code to generate palindrome numbers because we knew that that function was actually very fast and if we used that function then it would improve the code a lot and make it faster. Also, Abdallah was already using a fast algorithm for prime numbers so we knew that if we added the generate palindrome's function then it would improve the code by double and make it a lot faster for larger inputs. After we added the generate palindrome function that was used in Nikmal's improved code, we saw that the running time improved by a lot and it took 1 minute and 16 seconds for the last test case to run which was actually faster than Nikmal's improved code. The reason for this was because Abdallah had an algorithm which was the sieve_of_eratosthenes which made the code faster whereas Nikmal's prime function was more slower. Overall, what improved the code by a lot was that the same generate palindromes function was used from Nikmal's improved code and everything else was the same because we didn't know what else to change to make it run faster. The palindrome function that was used generates palindromes of a given length using the same way in Nikmal's improved code. If the palindrome is odd, it also generates palindromes by concatenating a number with it's reverse and with a middle digit. And the string palindrome was the same, the only difference that improved the code was the fact that the palindromes were being generated instead of going number by number.

Abdalla's Results

Test cases	Input (m)	Input (n)	Output	Correctness	Running time (s)
1	1	2_000	Total number of special numbers: 20 First 3 smallest special numbers: [2, 3, 5] Last 3 biggest special numbers: [797, 919, 929] Run time: 0.000000 seconds	The prime function was already very fast and he had used an algorithm in it. The smae palindrome function was used from Nikmal's improved code to generate palindrome	0.0 seconds

				s which improved it a lot.	
2	100	10_000	Total number of special numbers: 15 First 3 smallest special numbers: [101, 131, 151] Last 3 biggest special numbers: [797, 919, 929] Run time: 0.000000 seconds	The prime function was already very fast and he had used an algorithm in it. The smae palindrome function was used from Nikmal's improved code to generate palindromes which improved it a lot.	0.0 seconds
3	20_000	80_000	Total number of special numbers: 48 First 3 smallest special numbers: [30103, 30203, 30403] Last 3 biggest special numbers: [79397, 79697, 79997] Run time: 0.000996 seconds	The prime function was already very fast and he had used an algorithm in it. The smae palindrome function was used from Nikmal's improved code to generate palindromes which improved it a lot.	0.000996 seconds
4	100_000	2_000_000	Total number of special numbers: 190	The prime function was already	0.006977 seconds

			First 3 smallest special numbers: [1003001, 1008001, 1022201] Last 3 biggest special numbers: [1993991, 1995991, 1998991] Run time: 0.006977 seconds	very fast and he had used an algorithm in it. The smae palindrome function was used from Nikmal's improved code to generate palindromes which improved it a lot.	
5	2_000_000	9_000_000	Total number of special numbers: 327 First 3 smallest special numbers: [3001003, 3002003, 3007003] Last 3 biggest special numbers: [7985897, 7987897, 7996997] Run time: 0.013464 seconds	The prime function was already very fast and he had used an algorithm in it. The smae palindrome function was used from Nikmal's improved code to generate palindromes which improved it a lot.	0.013464 seconds
6	10_000_000	100_000_000	Total number of special numbers: 0 All 0 special numbers: [] Run time: 0.043368 seconds	The prime function was already very fast and he had used an algorithm in it. The smae palindrome function	0.043368 seconds

				was used from Nikmal's improved code to generate palindromes which improved it a lot.	
7	100_000_000	400_000_000	Total number of special numbers: 2704 First 3 smallest special numbers: [100030001, 100050001, 100060001] Last 3 biggest special numbers: [399737993, 399767993, 399878993] Run time: 0.356388 seconds	The prime function was already very fast and he had used an algorithm in it. The smae palindrome function was used from Nikmal's improved code to generate palindromes which improved it a lot.	0.356388 seconds
8	1_100_000_000	15_000_000_000	Total number of special numbers: 5474 First 3 smallest special numbers: [10000500001, 10000900001, 10001610001] Last 3 biggest special numbers: [14998289941, 14998589941, 14998689941]	The prime function was already very fast and he had used an algorithm in it. The smae palindrome function was used from Nikmal's improved code to generate palindromes which	5.195230 seconds

			Run time: 5.195230 seconds	improved it a lot.	
9	15_000_000_000	100_000_000_000	Total number of special numbers: 36568 First 3 smallest special numbers: [15001010051, 15002120051, 15002320051] Last 3 biggest special numbers: [99998189999, 99998989999, 99999199999] Run time: 66.619260 seconds	The prime function was already very fast and he had used an algorithm in it. The smae palindrome function was used from Nikmal's improved code to generate palindromes which improved it a lot.	66.619260 seconds
10	1	1_000_000_000_000	Total number of special numbers: 47995 First 3 smallest special numbers: [2, 3, 5] Last 3 biggest special numbers: [99998189999, 99998989999, 99999199999] Run time: 77.772752 seconds	The prime function was already very fast and he had used an algorithm in it. The smae palindrome function was used from Nikmal's improved code to generate palindromes which improved it a lot.	77.772752 seconds

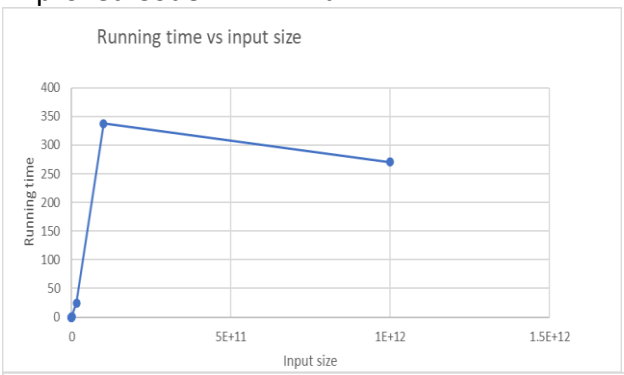
4. Compare the performance!

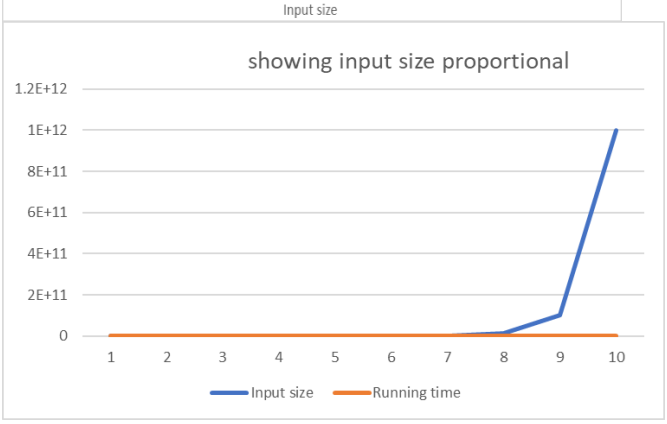
Time complexities and big-O notations

Improved Solutions	Time Complexities End time -	Big-O Notations Bench mark – how the time is like linear
Code 1 - Nikmal	**Exponential increase with the test cases**	**O(2^n)**
Code 2 - Abdalla	**Exponential increase with the test cases**	**O(2^n)**

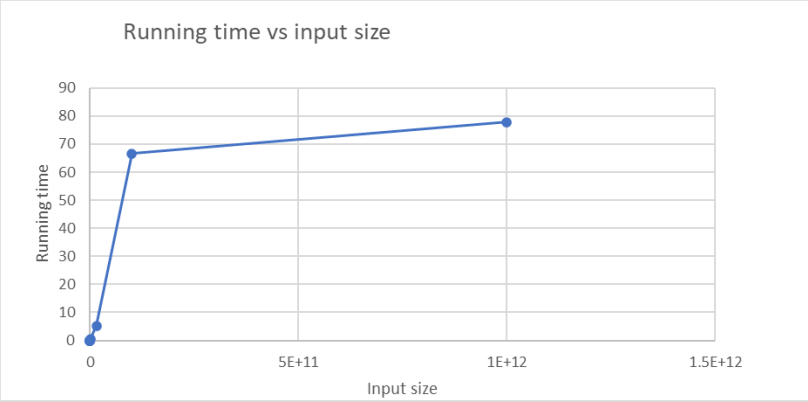
Running time graphs

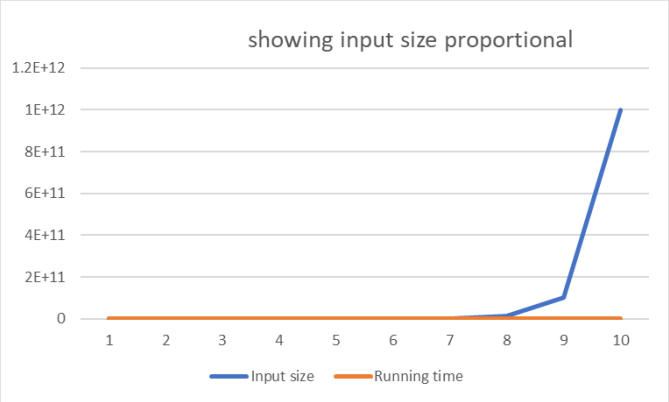
Improved Code 1 – Nikmal





Improved Code 2 – Adballa





5. Reflecting on teamwork!

Contribution mark

Name	ID	Task 1 (30%)	Task 2 (20%)	Task 3 (20%)	Task 4 (15%)	Task 5 (15%)	**Contribution mark (100%)**

Hernandez Upegui, Bryan (Group leader)	001305722	30%	20%	20%	15%	35%	100%
Niazi, Nikmal (Champion 1)	001311920	30%	20%	30%	15%	15%	100%
Luz, Louis Anthony (Champion 2)	001297211	30%	20%	20%	15%	15%	100%
Saleh, Abdalla	001241463	20%	20%	0%	0%	0%	40%
Alpoim ferreira do rosario, Alana taiana	001304790	20%	20%	0%	0%	0%	40%

Limitation discussion

We discuss the technical challenges, participation/engagement, collaboration, leadership, problem-solving skills, creativity and innovation, or communication dynamic topics. We agree on these following points:
We agree that we could have done more meeting apart from the group up in the lab, we could have done in Microsoft Teams or in Zoom.

Weekly journal

The figures, I'm refereeing to the screenshot for the chats bellow, in the WhatsApp group chat in the Appendix C - Evidence of team contribution section.

	Task note	**Status**
Week 1: from 3/02/2024 -17/02/2024		
Hernandez Upegui, Bryan (Group leader)	We started to contact everyone in the group (as you can see in figure1,2,3) and arranged meet in the lab for the next week, plus make the group chat in Microsoft Teams and in WhatsApp.	
Niazi, Nikmal	I had emailed every single student and created a WhatsApp group chat for us to communicate with each other. In the email I added the link for everyone to join. After a couple of days, everyone had joined the group chat and Bryan wanted to be group leader which we all agreed on. Bryan also created a Microsoft Teams group for us to the share the report and our code files for everyone to see and from this we were able to see what each member of the group were working on every week.	
Luz, Louis Anthony	Joined both the WhatsApp group and team's chat. Additionally, joined the shared doc report of the coursework after having asked for access to be able to read the report.	
Saleh, Abdalla	Established contact with everyone and began arranging meetups in the labs via Teams. WhatsApp group chat also created	
Alpoim ferreira do rosario, Alana taiana	Joined the WhatsApp and Microsoft Teams group and started do research for the course work.	
Week 2: from 12/02/2024 - 18/02/2024		

Hernandez Upegui, Bryan (Group leader)	On the 12/02/2024, I asked my teammates if i could be the group leader and all agree that I will become the leader of this coursework. (as seen in Figure 4)	
Niazi, Nikmal	In the second week, we decided to meet our group members and get to know each other, we discussed about the coursework and how to start and what to start on. We decided and also texted in the group chat (for the people who didn't come) that we all should start creating our main codes for the specification.	
Luz, Louis Anthony	We used the WhatsApp group to discuss roles and to ensure that we all understood the task. Therefore, we all decided to read the specification so that when we meet up next week, we have at least understood the task and started on it.	
Saleh, Abdalla	Arranged a meetup time (i personally could not make it so i made contact via WhatsApp) and talked about how to move forwards and agree to beginning the code creation individually	
Alpoim ferreira do rosario, Alana taiana	Discussed about the leader and organised a meeting in the lad to star with the process of coding.	
Week 3: from 19/02/2024 - 25/02/2024		
Hernandez Upegui, Bryan (Group leader)	In the start of the lab in the 20/02/2024, (as you can see in Figure 6) I massage the group chat that if anyone was in the lab, I was able to meet the people in the group (only Louis and Nikmal come to the lab to meet and talk about the coursework, Alana and Addalla didn't come, and we planned that in this week we decided that everyone had they code for task 1 for next week to decide for 2 best ones to improve and most people were sending it on team by the 25/02/2024(as you can see in figure 7,8,9,10)	
Niazi, Nikmal	In week 3, we were all still working on our individual code and tried to make it a bit better in the lab we would tell each other our progress.	
Luz, Louis Anthony	This week I met two of four groupmates at the tutorial; these being Bryan and Nikmal. We discussed with we understood the task and whether any of us have started writing our program. All of us said we have started but not completed yet.	
Saleh, Abdalla	Progression with each individuals code and concessions about said progression.	
Alpoim ferreira do rosario, Alana taiana	We organised a meeting in the lab but could not attend for personal reasons. I started the code process and defined the classes.	
Week 4: from 26/02/2024 - 03/03/2024		

Hernandez Upegui, Bryan (Group leader)	In this week we were trying to make the code run faster, the main person who was working on it was Nikmal, then Louiz and Bryan (as seen in figure 11, 12 &13)	
Niazi, Nikmal	In the fourth week, we had decided to improve my code and Abdallah's code. We had already finished improving my code so all that was left was to finish improving Abdallah's code.	
Luz, Louis Anthony	As for this week, we decided on who's code to improve being Nikmal's and Abdalla's code as they were running more of the tests given. Hence, we said for everyone to try and improve on them to make it run faster and till the last case being a trillion, which we will all give next week.	
Saleh, Abdalla	Unanimous decision to improve Nikmal's and my code due to performance. Nikmal's code had been improved and all that was left was mine	
Alpoim ferreira do rosario, Alana taiana	Improved my code posted my final code on Microsoft teams.	
Week 5: from 04/03/2024 - 12/03/2024		
Hernandez Upegui, Bryan (Group leader)	We continue to improve the 2 code that were Adballa and Nikmal (as seen in figure 14, 15). We started to work on the report, mainly Nikmal, Loui and Bryan. The other 2 team mate didn't respond back (seen in figure 15) ask them if they could fill in they part of the report, message send on the 10/03/2024, no respond in the 12/03/2024, I message again in the group chat asking Alana and Abdalla, to do they part of the report if not I will lower they participation percentage (seen in figure 16),this time Abdalla reponed but Alana didn't respond. As seen in Communication logs – Microsoft word, i have put screenshots of the information of the document of the report that show who has been editing the report, this is evidence that, me Bryan Hernanadez Upegui, Nikmal, Louis has been working on the report Seens 6 of February to the 13 of march	
Niazi, Nikmal	Finally, in the last week, our group had basically finished improving both codes and we also each had a main code and solution which was different. In this week, we came to the lab and talked about finishing the report. Each person had to do their own part for each task. individually.	
Luz, Louis Anthony	Having finished the improvements on the 2 codes, we all decided to work on the report with everyone doing their individual parts and discussing who will do the shared parts like task 4 or 2.	
Saleh, Abdalla	Individually working on report after finishing improving the 2 codes and verifying everyone had their main code finished as well.	
Alpoim ferreira do rosario, Alana taiana	On 05/03/24 I went to the lab were Bryan, Nikmal and Louis were present. Uploaded my zip file of my code on Microsoft teams on the same day.	

Week 6: from 12/03/2024 - 17/03/2024		
Hernandez Upegui, Bryan (Group leader)	no respond in the 12/03/2024, I message again in the group chat asking Alana and Abdalla, to do their part of the report if not I will lower, their participation percentage (seen in figure 16),this time Abdalla reponed but Alana didn't respond. on the 14/03/2024 Abdalla ask me for access for the document to do his part and I gave him access as seen in figure 17, then he told me he did his part but when I check he did task1 but not his part in task 2, I send a message telling him to do his task 2 part no respond for that day or the next, in all this time Alana didn't respond to my previous messages and on the 15 Alana finally responded which asking me about the report as you see in the message in figure 18 I told her about our previous discussion and messages and told her how to access the share report and gave my final warning to Abdalla and Alana for them to o they part by Friday after that I will not take any more work as I already told them a few days before to do it by Thursday but no they didn't or message or communicated to me anything so finally by Friday and on Saturday the 16/03/2024 I check and they did they part. In these few days I been working on checking all the report, doing task 1 comparing 2 code of the student my one Bryan and Louis code and as all the people finally completed task 2 I was able to compare, they result of they own test cases and complexity. Also finalizing any little thighs on the report so that it can be complete before the deadline and can be submitted before.	
Niazi, Nikmal	In the last week of working on the report, I had finished creating the report and double checked the report to see if there were any mistakes made or any changes to be made. For example, if there were any grammar or spelling mistakes.	
Luz, Louis Anthony	Within this week, just completed the report.	
Saleh, Abdalla		
Alpoim ferreira do rosario, Alana taiana	Worked on the report.	

References

Big O - Step into Wix Studio | The web platform for agencies and enterprises (youtube.com)

Check later for the prime number part if
math - efficiently finding prime numbers in python - Stack Overflow use -
Tuan Vuong, COMP1819ADS, (2022), GitHub repository, prime numbers
COMP1819ADS/Lab_02/ANS_04_isPrime_v1.py at main · vptuan/COMP1819ADS · GitHub

Stack Overflow,
math - efficiently finding prime numbers in python - Stack Overflow

Special prime numbers - GeeksforGeeks - check the 3 or 4 and select python3 and the prime number

Tuan Vuong, COMP1819ADS, (2022), GitHub repository, taking time
COMP1819ADS/Lab_02/00_time_taken.py at main · vptuan/COMP1819ADS · GitHub

6 Best Ways To Check If Number Is Prime In Python - (pythonpool.com) - Was used to find fast ways to check for prime numbers

python - How to generate a list of palindrome numbers within a given range? - Stack Overflow - This was used to help find a way to improve the palindrome function for the improved code.

Tuan Vuong, COMP1819ADS, (2022), GitHub repository,
https://github.com/vptuan/COMP1819ADS

W3Schools (2019). *Python Tutorial*. [online] W3schools.com. Available at: https://www.w3schools.com/python/ [Accessed 9 Feb. 2024].

Prime Numbers: What and Why – The Math Doctors - use to understand more on what to do to get the special number

Appendix A.1 - Proposed solution 1 - 6

Student 1 - Bryan's Code:

```
1. import time
```

```
    def is_prime(n):
        if n < 2:
            return False
        for i in range(2, int(n**0.5) + 1):
            if n % i == 0:
                return False
        return True


    def is_palindrome(n):

        return str(n) == str(n)[::-1]


    def find_special_numbers(m, n):

        special_numbers = []
        for num in range(m, n + 1):
            if is_prime(num) and is_palindrome(num):
                special_numbers.append(num)
        return special_numbers


    def display_special_numbers(special_numbers):
        total = len(special_numbers)
        if total < 6:
            print("Total: Special Numbers =", total)
            print("List of special numbers =", special_numbers)
        else:
            print("Total: special Numbers =", total)
            print("List of special numbers =", special_numbers[:3] + special_numbers[-3:])


    def main():
        m = int(input("Enter the lower limit (m): "))
        n = int(input("Enter the upper limit (n): "))
        if m > n or m < 1 or n < 1:
            print("Invalid input. Please enter valid numbers.")
        else:
            start_time = time.time()
            special_numbers = find_special_numbers(m, n)
            display_special_numbers(special_numbers)
            end_time = time.time()
            print("Execution time: {:.6f} seconds".format(end_time - start_time))


    if __name__ == "__main__":
        main()

2.
```

Student 2 – Nikmal's Code:

```
1.  import timeit as t
2.  import math
3.
4.  def find_special_numbers(m, n):
5.      special_numbers = []
6.      for num in range(m, n + 1):
7.          if is_palindromic(num) and is_prime(num):
```

```
8.              special_numbers.append(num)
9.      return special_numbers
10.
11. def is_prime(n):
12.     a = 2
13.     while a <= math.sqrt(n):
14.         if n % a < 1:
15.             return False
16.         a = a + 1
17.     return n > 1
18.
19. def is_palindromic(n):
20.     string_n = str(n)
21.     return string_n == string_n[::-1]
22.
23. def display_special_numbers(special_numbers):
24.     if len(special_numbers) < 6:
25.         print("All special numbers:", special_numbers)
26.     else:
27.         print("First three smallest special numbers:", special_numbers[:3])
28.         print("Last three largest special numbers:", special_numbers[-3:])
29.
30. def main():
31.     m = int(input("Please enter a positive number (m): "))
32.     n = int(input("Please enter a bigger positive number (n): "))
33.     if m < 1 or n < 1:
34.         print("You can't enter a negative number")
35.     if m > n:
36.         print("M must be smaller than N")
37.     start = t.default_timer()
38.     special_numbers = find_special_numbers(m, n)
39.     print("Total number of special numbers:", len(special_numbers))
40.     display_special_numbers(special_numbers)
41.     end = t.default_timer()
42.     print(f"Run Time: {end - start} seconds")
43.
44. if __name__ == "__main__":
45.     main()
```

Student 3 – Louis Anthony's Solution:

```
import time  # Import the time module for measuring execution time

class special_number:  # Define a class named special_number
    def __init__(self, Prime, Palindrome):
        self.prime_num = Prime
        self.palindrome = Palindrome

    def is_Prime(self, x):  # Method to check if a number is prime
        if x < 2:
```

```
            return False
        if x == 2:
            return True
        if x % 2 == 0:
            return False
        for i in range(3, int(x ** 0.5) + 1, 2):
            if x % i == 0:
                return False
        return True

    def is_Palindromic(self, x):  # Method to check if a number is palindromic
        num = str(x)
        return num == num[::-1]

m = int(input('Enter lower range number:'))  # Get lower range input from the user
n = int(input('Enter higher range number:'))  # Get higher range input from the user

start_time = time.time()  # Record the start time for performance measurement
special_numbers = [] # empty set

# used Chatgpt to help for this code
while m <= n:
    if special_number(m, m).is_Prime(m) and special_number(m, m).is_Palindromic(m):
        special_numbers.append(m)
    m += 1

print('There are', len(special_numbers), 'numbers')

if len(special_numbers) > 6:  # if statement on the amount of special numbers
    print('The first 3 special numbers are:', special_numbers[:3]) # this string :3
finds the first three in the empty list
    print('The last 3 special numbers are:', special_numbers[-3:]) # this string -3:
finds the last three in the empty list
else:
    print('The special numbers are', special_numbers)

end_time = time.time()  # Record the end time for performance measurement
time_taken = end_time - start_time
print('Time taken:', time_taken, 'secs')  # Print the time taken for execution
```

Student 4 – Abdalla's Solution:

```
1.  import time
2.
3.  def is_prime(num, primes):
4.      if num < 2:
5.          return False
6.      for prime in primes:
7.          if prime * prime > num:
8.              return True
9.          if num % prime == 0:
10.             return False
11.     return True
```

```
12.
13. def sieve_of_eratosthenes_segmented(limit):
14.     sieve = [True] * (limit + 1)
15.     sieve[0] = sieve[1] = False
16.
17.     for num in range(2, int(limit**0.5) + 1):
18.         if sieve[num]:
19.             for multiple in range(num*num, limit + 1, num):
20.                 sieve[multiple] = False
21.
22.     primes = [num for num in range(2, limit + 1) if sieve[num]]
23.
24.     return primes
25.
26. def is_palindrome(num):
27.     return str(num) == str(num)[::-1]
28.
29. def find_special_numbers(m, n):
30.     start_time = time.time()
31.
32.     sqrt_n = int(n**0.5) + 1
33.     primes = sieve_of_eratosthenes_segmented(sqrt_n)
34.
35.     special_numbers = [num for num in range(m, n + 1) if is_prime(num, primes) and
    is_palindrome(num)]
36.
37.     if len(special_numbers) <= 5:
38.         print(f"All {len(special_numbers)} special numbers: {special_numbers}")
39.     else:
40.         print(f"First 3 smallest special numbers: {special_numbers[:3]}")
41.         print(f"Last 3 biggest special numbers: {special_numbers[-3:]}")
42.     print(f"Total number of special numbers: {len(special_numbers)}")
43.
44.     end_time = time.time()
45.     print(f"Runtime: {end_time - start_time:.6f} seconds")
46.
47. if __name__ == "__main__":
48.     m = int(input("Enter the smaller number (m): "))
49.     n = int(input("Enter the larger number (n): "))
50.
51.     if m >= n:
52.         print("Error: m should be smaller than n.")
53.     else:
54.         find_special_numbers(m, n)
```

Student 5 – Alana taiana's Solution:

```
import time


def is_prime(n):
    if n < 2:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    sqrt_n = int(n ** 0.5) + 1
    for i in range(3, sqrt_n, 2):
```

```
        if n % i == 0:
            return False
    return True


def is_palindrome(n):
    return str(n) == str(n)[::-1]


def find_special_numbers(m, n):
    special_numbers = [num for num in range(m, n + 1) if is_prime(num) and is_palindrome(num)]
    return special_numbers


def display_special_numbers(special_numbers, m, n):
    if len(special_numbers) < 6:
        print("All special numbers between {} and {} are: {}".format(m, n, special_numbers))
    else:
        print("First 3 smallest special numbers between {} and {} are: {}".format(m, n, special_numbers[:3]))
        print("Last 3 biggest special numbers between {} and {} are: {}".format(m, n, special_numbers[-3:]))
    print("Total number of special numbers between {} and {} is: {}".format(m, n, len(special_numbers)))


def main():
    try:
        m = int(input("Enter the lower limit (m): "))
        n = int(input("Enter the upper limit (n): "))
        if m < 0 or n < 0 or m > n:
            print("Invalid input. Please enter positive numbers with m < n.")
            return
    except ValueError:
        print("Invalid input. Please enter valid integers.")
        return

    start_time = time.time()
    special_numbers = find_special_numbers(m, n)
    end_time = time.time()

    display_special_numbers(special_numbers, m, n)
    print("Elapsed time:", end_time - start_time, "seconds")


if __name__ == "__main__":
    main()
```

Appendix B - Test cases for correctness

Student 1 – Nikmal's Full Improved Code:

```
1.  import timeit as t
2.  import math
3.
4.  def find_special_numbers(m, n):
5.      special_numbers = []
6.      for length in range(len(str(m)), len(str(n)) + 1):
7.          palindromes = generate_palindromes(length)
8.          for num in palindromes:
```

```
            if m <= num <= n and is_prime(num):
                special_numbers.append(num)
    return special_numbers

def is_palindromic(n):
    return str(n) == str(n)[::-1]

def generate_palindromes(length):
    if length == 1:
        return range(1, 10)
    palindromes = []
    if length % 2 == 0:
        half_length = length // 2
        for num in range(10 ** (half_length - 1), 10 ** half_length):
            s = str(num)
            palindromes.append(int(s + s[::-1]))
    else:
        half_length = (length - 1) // 2
        for num in range(10 ** (half_length - 1), 10 ** half_length):
            s = str(num)
            for mid_digit in range(10):
                palindromes.append(int(s + str(mid_digit) + s[::-1]))
    return palindromes

def is_prime(n):
    if n <= 1:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    max_divisor = math.floor(math.sqrt(n))
    for d in range(3, max_divisor + 1, 2):
        if n % d == 0:
            return False
    return True

def display_special_numbers(special_numbers):
    if len(special_numbers) < 6:
        print("All special numbers:", special_numbers)
    else:
        print("First three smallest special numbers", special_numbers[:3])
        print("Last three largest special numbers", special_numbers[-3:])

def main():
    m = int(input("Please enter a positive number (m): "))
    n = int(input("Please enter a bigger positive number (n): "))
    if m < 1 or n < 1 or m > n:
        print("You can't enter a negative number and M must be smaller than N")
    start = t.default_timer()
    special_numbers = find_special_numbers(m, n)
    print("Total number of special numbers:", len(special_numbers))
    display_special_numbers(special_numbers)
    end = t.default_timer()
    print(f"Run time: {end - start} seconds")

if __name__ == "__main__":
    main()
```

Student 2 – Abdallah's Full Improved Code:

```python
import time

def is_prime(num, primes):
    if num < 2:
        return False
    for prime in primes:
        if prime * prime > num:
            return True
        if num % prime == 0:
            return False
    return True

def sieve_of_eratosthenes_segmented(limit):
    sieve = [True] * (limit + 1)
    sieve[0] = sieve[1] = False

    for num in range(2, int(limit**0.5) + 1):
        if sieve[num]:
            for multiple in range(num*num, limit + 1, num):
                sieve[multiple] = False

    primes = [num for num in range(2, limit + 1) if sieve[num]]

    return primes

def is_palindrome(num):
    return str(num) == str(num)[::-1]

def generate_palindromes(length):
    if length == 1:
        return range(1, 10)
    palindromes = []
    if length % 2 == 0:
        half_length = length // 2
        for num in range(10 ** (half_length - 1), 10 ** half_length):
            s = str(num)
            palindromes.append(int(s + s[::-1]))
    else:
        half_length = (length - 1) // 2
        for num in range(10 ** (half_length - 1), 10 ** half_length):
            s = str(num)
            for mid_digit in range(10):
                palindromes.append(int(s + str(mid_digit) + s[::-1]))
    return palindromes

def find_special_numbers(m, n):
    start_time = time.time()

    sqrt_n = int(n ** 0.5) + 1
    primes = sieve_of_eratosthenes_segmented(sqrt_n)

    max_palindrome_length = len(str(n))
    palindromes = []
    for length in range(1, max_palindrome_length + 1):
```

```
55.         palindromes.extend(generate_palindromes(length))
56.     palindromes = [palindrome for palindrome in palindromes if m <= palindrome <= n]
57.
58.     special_numbers = [num for num in palindromes if is_prime(num, primes)]
59.
60.     if len(special_numbers) <= 5:
61.         print(f"All {len(special_numbers)} special numbers: {special_numbers}")
62.     else:
63.         print(f"First 3 smallest special numbers: {special_numbers[:3]}")
64.         print(f"Last 3 biggest special numbers: {special_numbers[-3:]}")
65.     print(f"Total number of special numbers: {len(special_numbers)}")
66.
67.     end_time = time.time()
68.     print(f"Run time: {end_time - start_time:.6f} seconds")
69.
70. if __name__ == "__main__":
71.     m = int(input("Enter the smaller number (m): "))
72.     n = int(input("Enter the larger number (n): "))
73.
74.     if m >= n:
75.         print("Error: m should be smaller than n.")
76.     else:
77.         find_special_numbers(m, n)
```

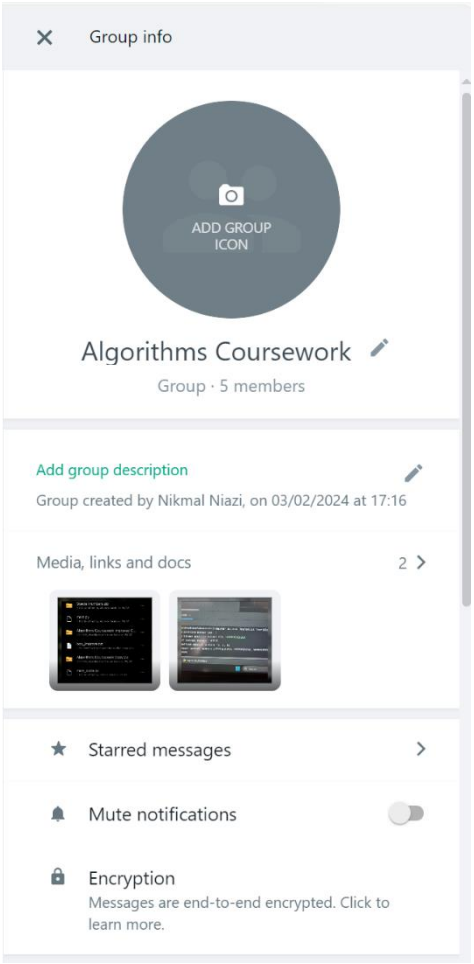
Appendix C - Evidence of team contribution

Communication logs – WhatsApp (Group chat)

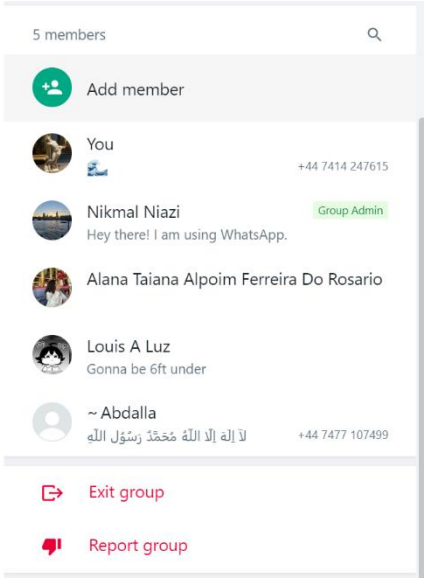
I understand that WhatsApp may be informal may to communicate for work, but in the scenery, it was the best choice as most people use the app as it is popular, and it was the most convenience and easiest way to communicate with all the people in the group. Also, the pictures are label as they are being reference in the weekly Journal to provide evidence for what it is stated in the journal.

These are screenshots (the screenshots were taken by Bryan Hernanez Upegui (group leader)) are the information of the group chat that was made by Nikmal and all the people of the coursework are in the group chat.(a seen in Figure 1 &2)

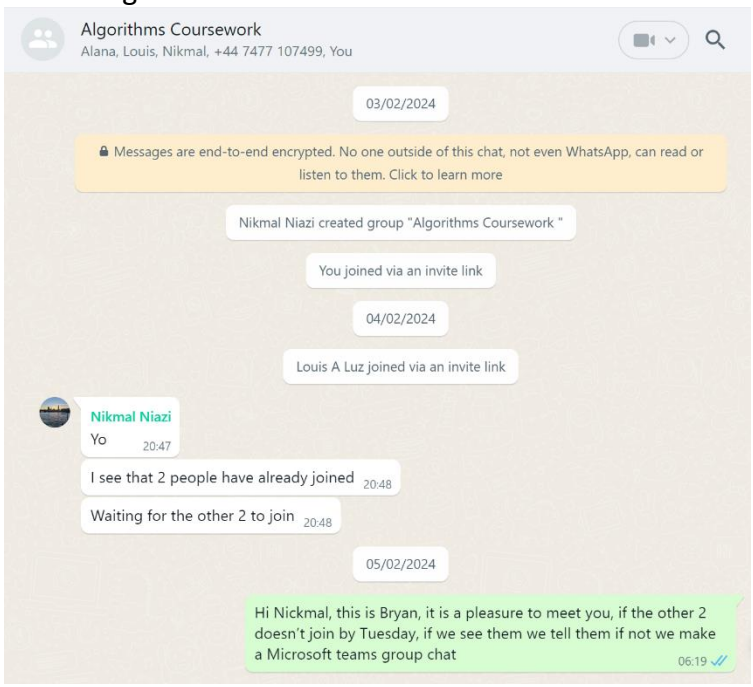
- Figure 1


Group info
ADD GROUP ICON
Algorithms Coursework
Group · 5 members
Add group description
Group created by Nikmal Niazi, on 03/02/2024 at 17:16
Media, links and docs
2
Starred messages
Mute notifications
Encryption
Messages are end-to-end encrypted. Click to learn more.

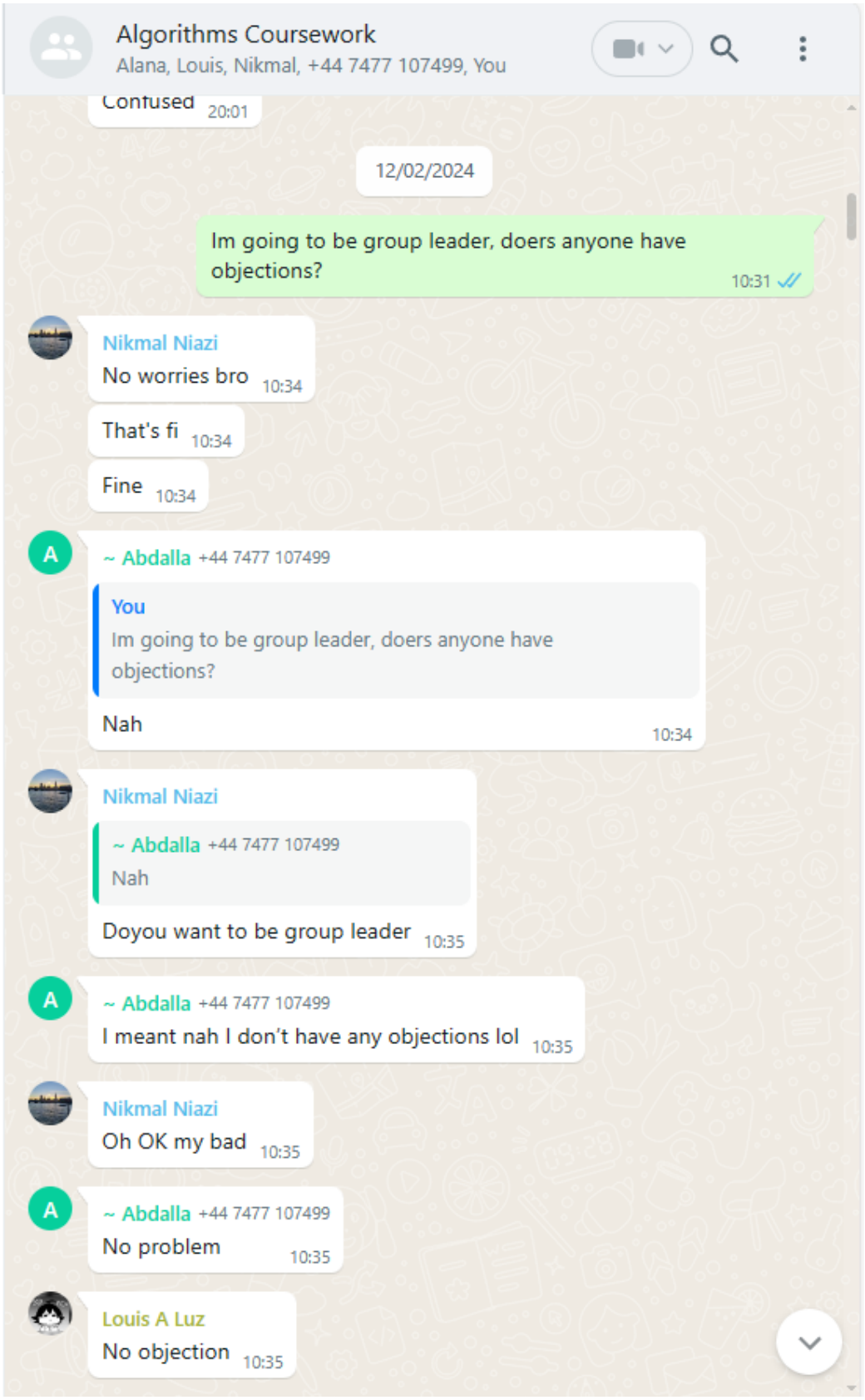
- Figure 2


5 members
Add member
You
+44 7414 247615
Nikmal Niazi
Group Admin
Hey there! I am using WhatsApp.
Alana Taiana Alpoim Ferreira Do Rosario
Louis A Luz
Gonna be 6ft under
~ Abdalla
لَا إِلَهَ إِلَّا اللَّهُ مُحَمَّدٌ رَسُولُ اللَّهِ
+44 7477 107499
Exit group
Report group

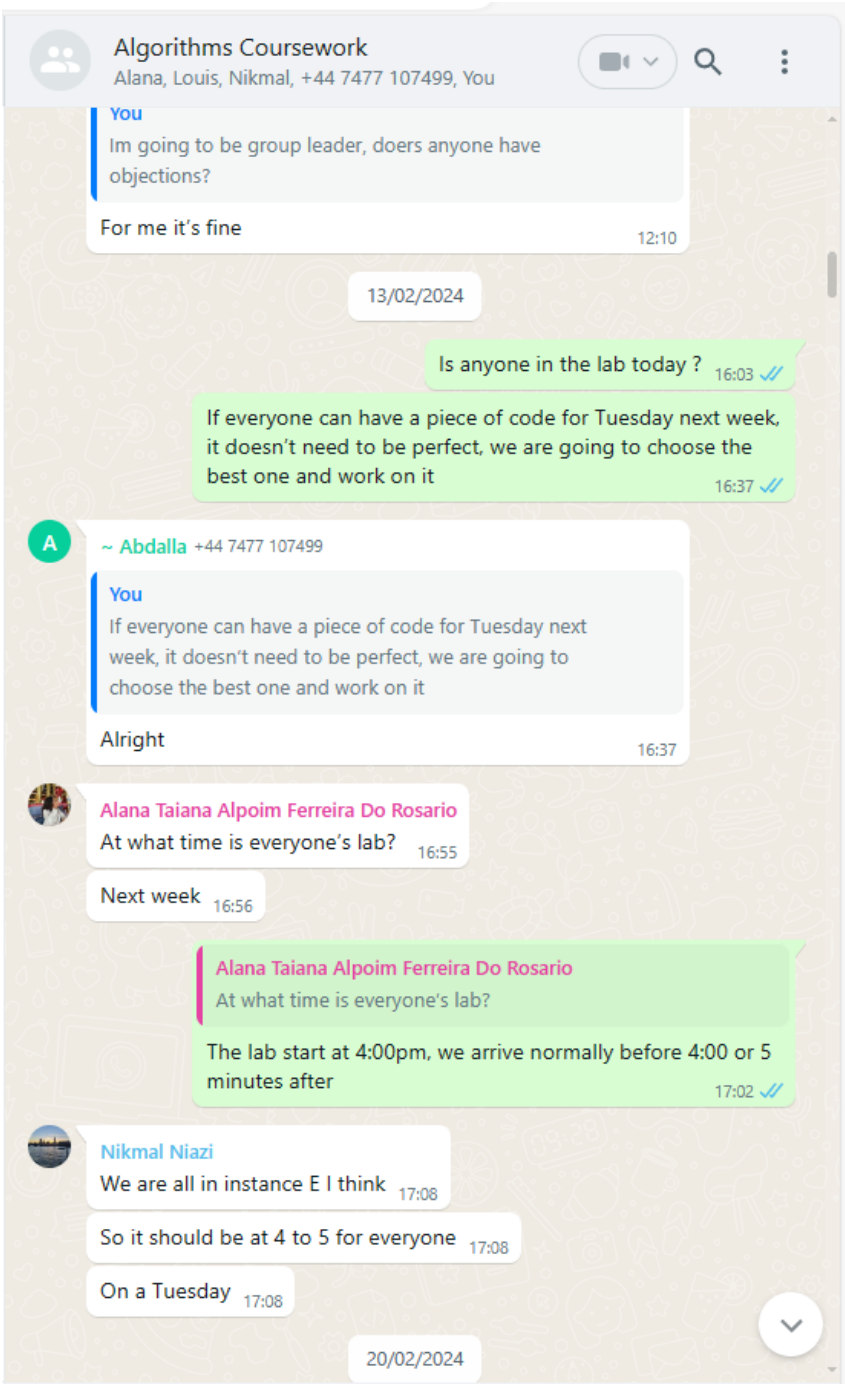
- Figure 3



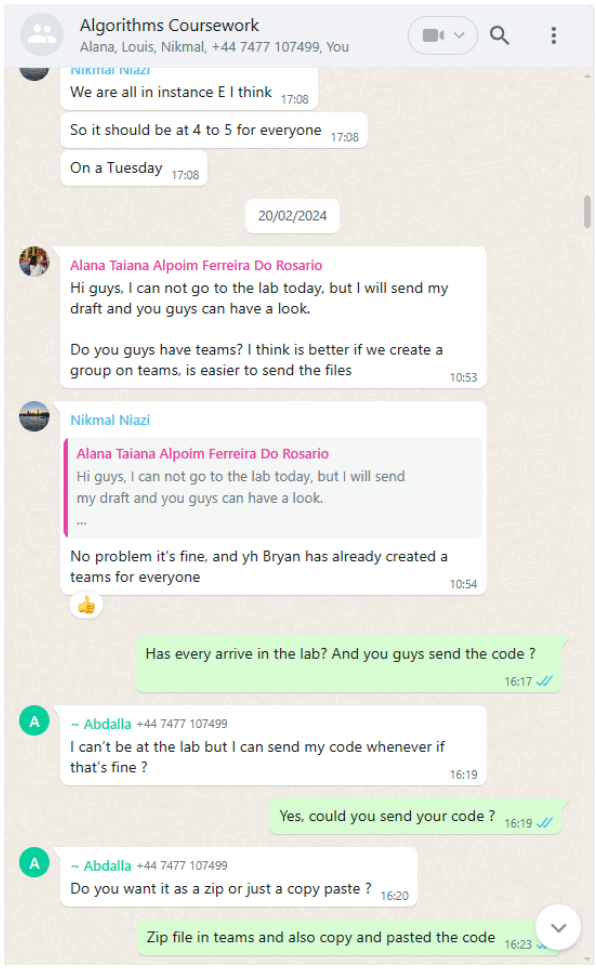
- Figure 4


Algorithms Coursework
Alana, Louis, Nikmal, +44 7477 107499, You
Confused 20:01
12/02/2024
Im going to be group leader, doers anyone have objections? 10:31
Nikmal Niazi
No worries bro 10:34
That's fi 10:34
Fine 10:34
~ Abdalla +44 7477 107499
You
Im going to be group leader, doers anyone have objections?
Nah 10:34
Nikmal Niazi
~ Abdalla +44 7477 107499
Nah
Doyou want to be group leader 10:35
~ Abdalla +44 7477 107499
I meant nah I don't have any objections lol 10:35
Nikmal Niazi
Oh OK my bad 10:35
~ Abdalla +44 7477 107499
No problem 10:35
Louis A Luz
No objection 10:35

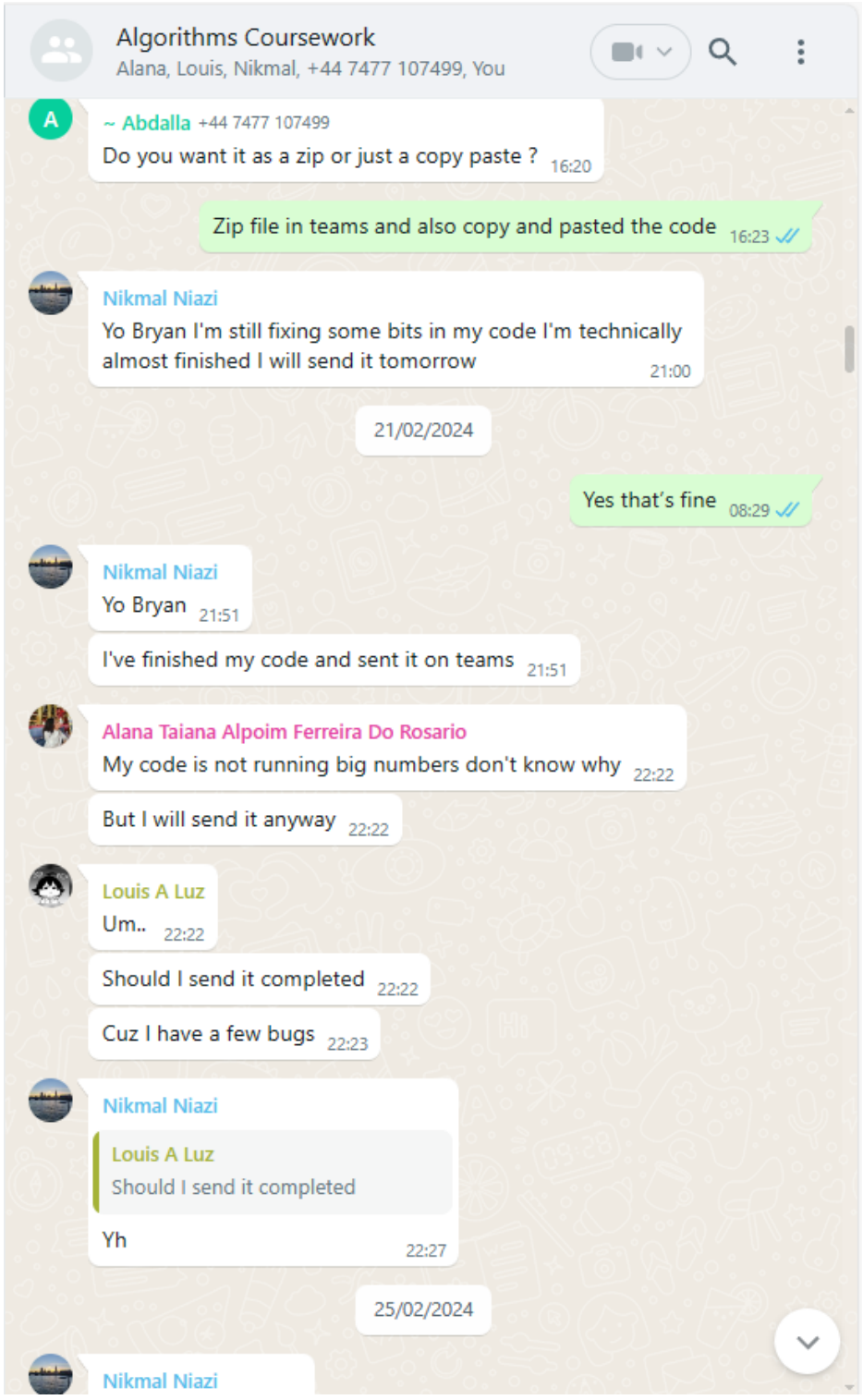
- Figure 5


Algorithms Coursework
Alana, Louis, Nikmal, +44 7477 107499, You
You
Im going to be group leader, doers anyone have objections?
For me it's fine
12:10
13/02/2024
Is anyone in the lab today ?
16:03
If everyone can have a piece of code for Tuesday next week, it doesn't need to be perfect, we are going to choose the best one and work on it
16:37
~ Abdalla +44 7477 107499
You
If everyone can have a piece of code for Tuesday next week, it doesn't need to be perfect, we are going to choose the best one and work on it
Alright
16:37
Alana Taiana Alpoim Ferreira Do Rosario
At what time is everyone's lab?
16:55
Next week
16:56
Alana Taiana Alpoim Ferreira Do Rosario
At what time is everyone's lab?
The lab start at 4:00pm, we arrive normally before 4:00 or 5 minutes after
17:02
Nikmal Niazi
We are all in instance E I think
17:08
So it should be at 4 to 5 for everyone
17:08
On a Tuesday
17:08
20/02/2024

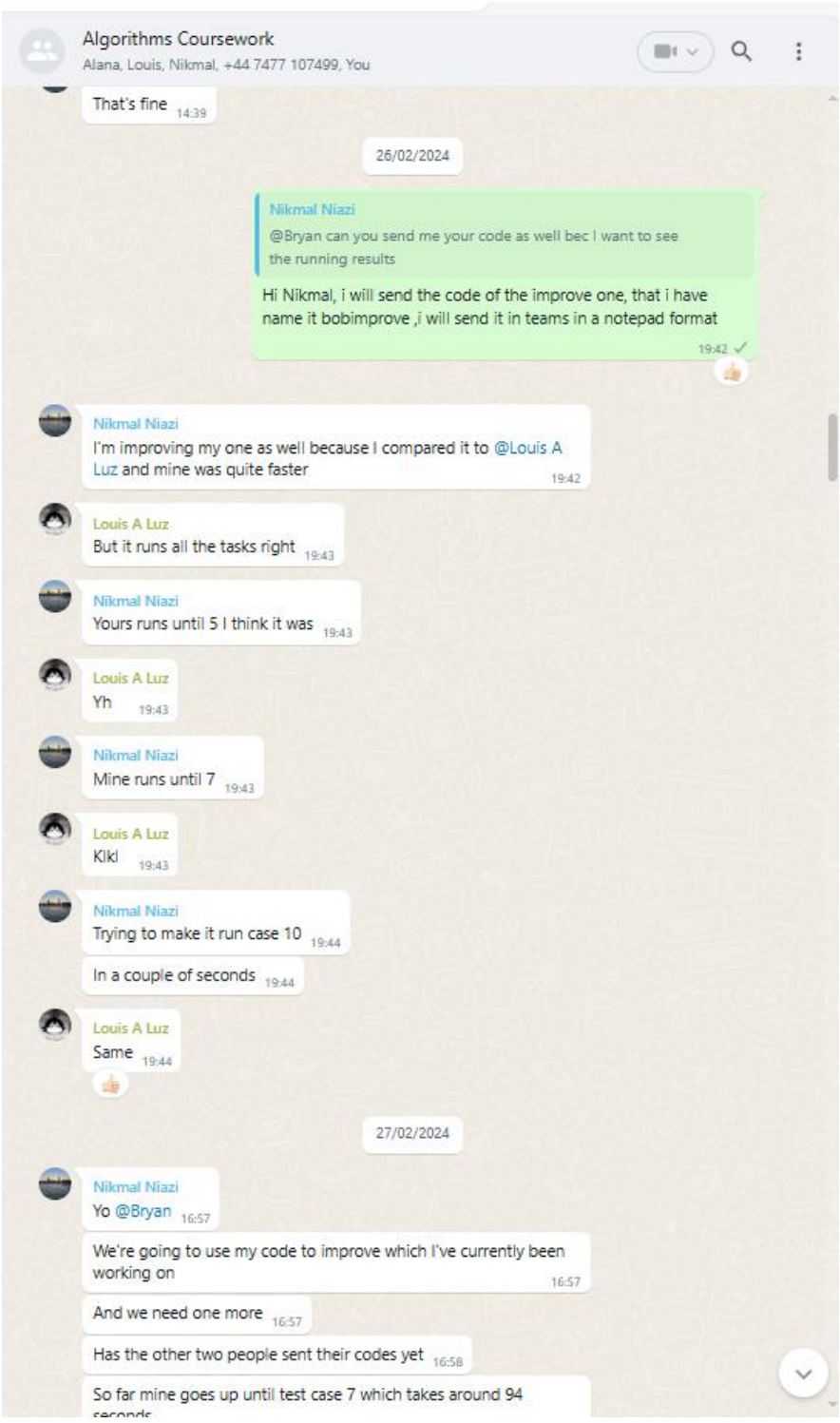
- Figure 6


Algorithms Coursework
Alana, Louis, Nikmal, +44 7477 107499, You
We are all in instance E I think 17:08
So it should be at 4 to 5 for everyone 17:08
On a Tuesday 17:08
20/02/2024
Alana Taiana Alpoim Ferreira Do Rosario
Hi guys, I can not go to the lab today, but I will send my draft and you guys can have a look.
Do you guys have teams? I think is better if we create a group on teams, is easier to send the files 10:53
Nikmal Niazi
Alana Taiana Alpoim Ferreira Do Rosario
Hi guys, I can not go to the lab today, but I will send my draft and you guys can have a look.
...
No problem it's fine, and yh Bryan has already created a teams for everyone 10:54
Has every arrive in the lab? And you guys send the code ? 16:17
~ Abdalla +44 7477 107499
I can't be at the lab but I can send my code whenever if that's fine ? 16:19
Yes, could you send your code ? 16:19
~ Abdalla +44 7477 107499
Do you want it as a zip or just a copy paste ? 16:20
Zip file in teams and also copy and pasted the code 16:23

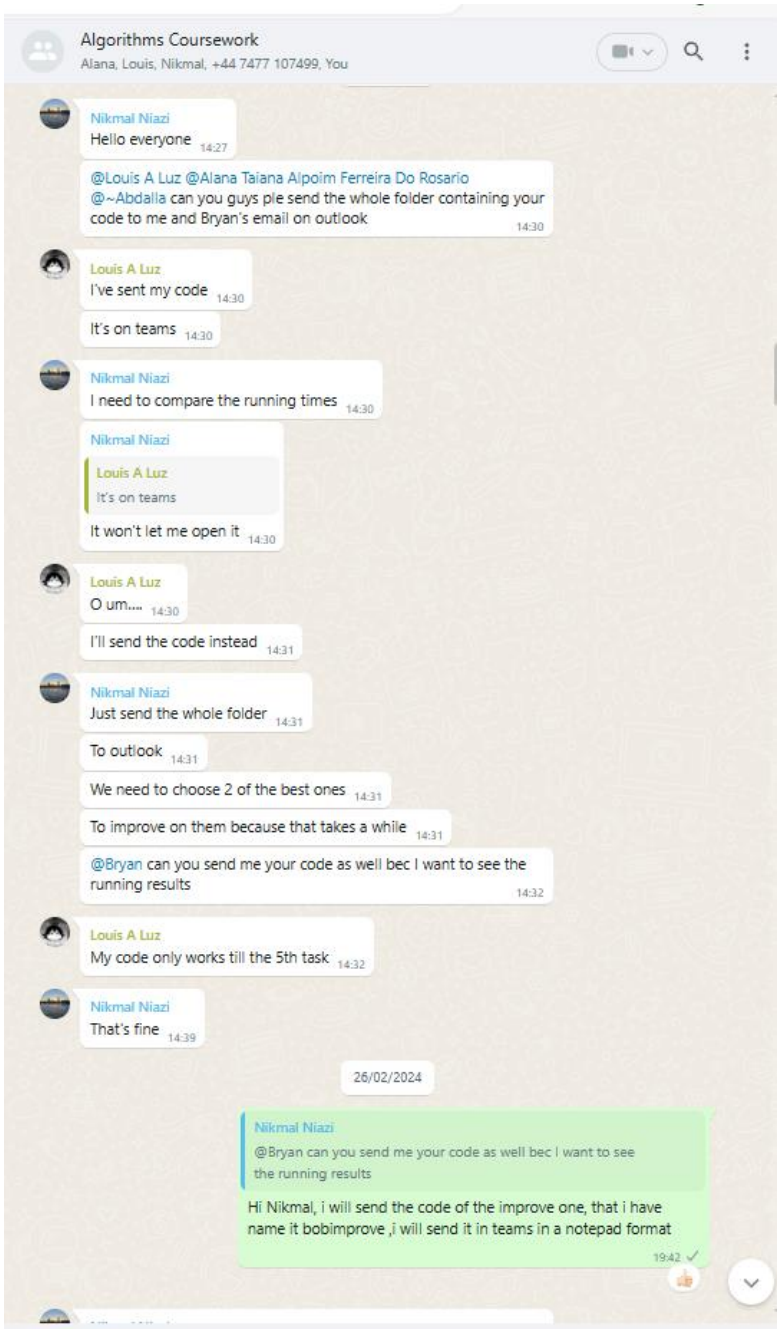
- Figure7


Algorithms Coursework
Alana, Louis, Nikmal, +44 7477 107499, You
~ Abdalla +44 7477 107499
Do you want it as a zip or just a copy paste ? 16:20
Zip file in teams and also copy and pasted the code 16:23
Nikmal Niazi
Yo Bryan I'm still fixing some bits in my code I'm technically almost finished I will send it tomorrow 21:00
21/02/2024
Yes that's fine 08:29
Nikmal Niazi
Yo Bryan 21:51
I've finished my code and sent it on teams 21:51
Alana Taiana Alpoim Ferreira Do Rosario
My code is not running big numbers don't know why 22:22
But I will send it anyway 22:22
Louis A Luz
Um.. 22:22
Should I send it completed 22:22
Cuz I have a few bugs 22:23
Nikmal Niazi
Louis A Luz
Should I send it completed
Yh 22:27
25/02/2024
Nikmal Niazi

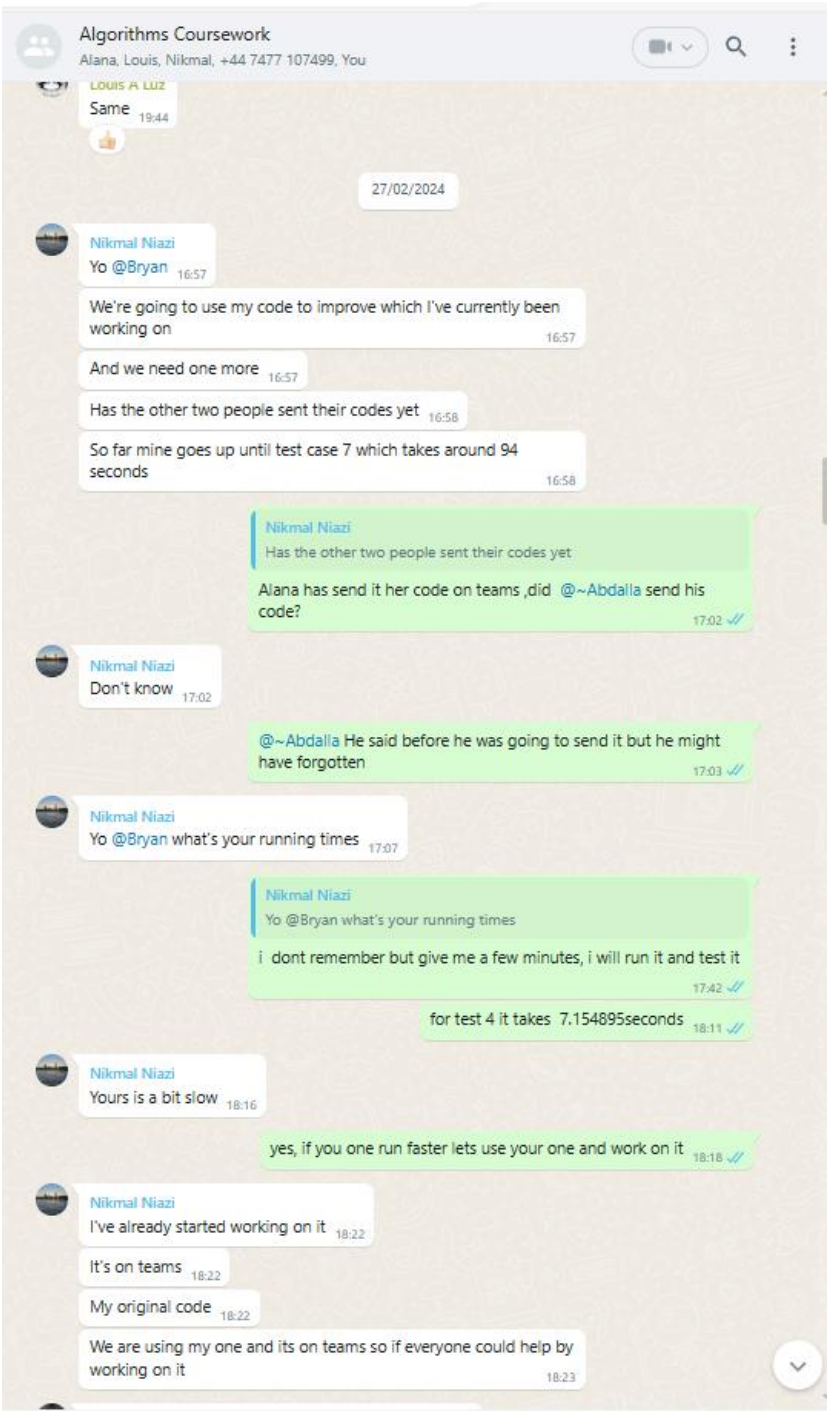
- Figure 8


Algorithms Coursework
Alana, Louis, Nikmal, +44 7477 107499, You
That's fine 14:39
26/02/2024
Nikmal Niazi
@Bryan can you send me your code as well bec I want to see the running results
Hi Nikmal, i will send the code of the improve one, that i have name it bobimprove ,i will send it in teams in a notepad format
19:42
Nikmal Niazi
I'm improving my one as well because I compared it to @Louis A Luz and mine was quite faster 19:42
Louis A Luz
But it runs all the tasks right 19:43
Nikmal Niazi
Yours runs until 5 I think it was 19:43
Louis A Luz
Yh 19:43
Nikmal Niazi
Mine runs until 7 19:43
Louis A Luz
Klkl 19:43
Nikmal Niazi
Trying to make it run case 10 19:44
In a couple of seconds 19:44
Louis A Luz
Same 19:44
27/02/2024
Nikmal Niazi
Yo @Bryan 16:57
We're going to use my code to improve which I've currently been working on 16:57
And we need one more 16:57
Has the other two people sent their codes yet 16:58
So far mine goes up until test case 7 which takes around 94

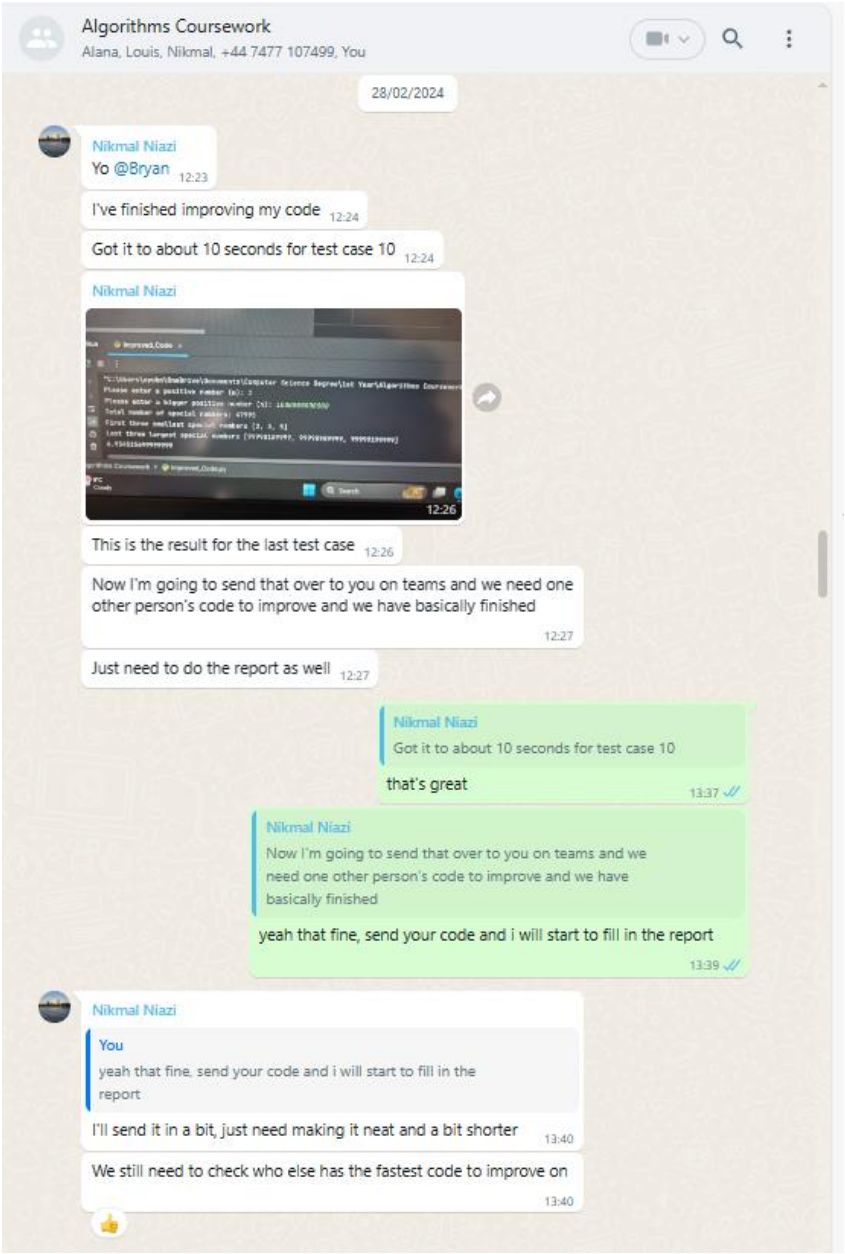
- Figure 9


Algorithms Coursework
Alana, Louis, Nikmal, +44 7477 107499, You
Nikmal Niazi
Hello everyone 14:27
@Louis A Luz @Alana Taiana Alpoim Ferreira Do Rosario @~Abdalla can you guys ple send the whole folder containing your code to me and Bryan's email on outlook 14:30
Louis A Luz
I've sent my code 14:30
It's on teams 14:30
Nikmal Niazi
I need to compare the running times 14:30
Nikmal Niazi
Louis A Luz
It's on teams
It won't let me open it 14:30
Louis A Luz
O um.... 14:30
I'll send the code instead 14:31
Nikmal Niazi
Just send the whole folder 14:31
To outlook 14:31
We need to choose 2 of the best ones 14:31
To improve on them because that takes a while 14:31
@Bryan can you send me your code as well bec I want to see the running results 14:32
Louis A Luz
My code only works till the 5th task 14:32
Nikmal Niazi
That's fine 14:39
26/02/2024
Nikmal Niazi
@Bryan can you send me your code as well bec I want to see the running results
Hi Nikmal, i will send the code of the improve one, that i have name it bobimprove ,i will send it in teams in a notepad format
19:42

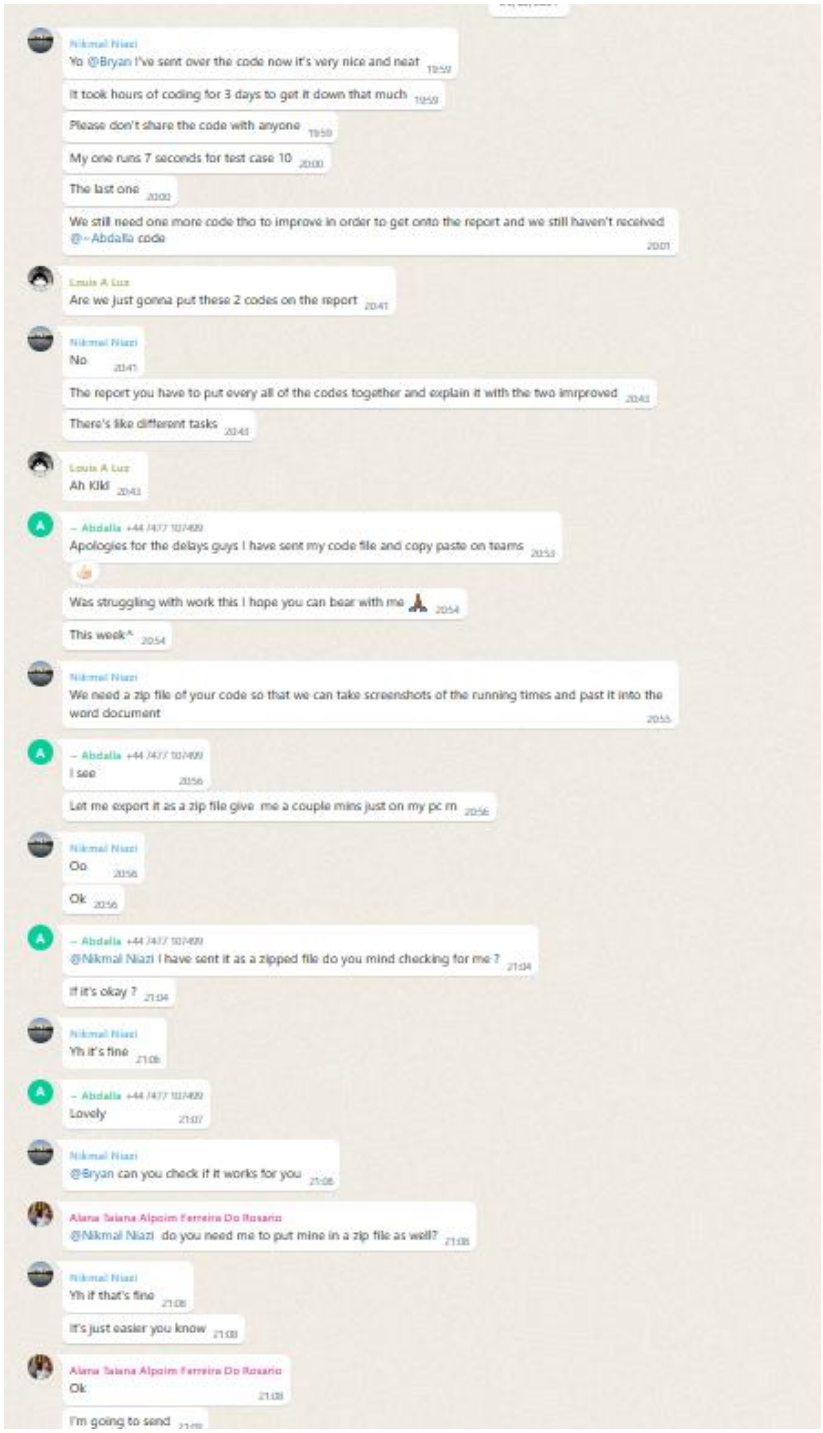
- Figure 10


Algorithms Coursework
Alana, Louis, Nikmal, +44 7477 107499, You
Louis A Luz
Same 19:44
27/02/2024
Nikmal Niazi
Yo @Bryan 16:57
We're going to use my code to improve which I've currently been working on 16:57
And we need one more 16:57
Has the other two people sent their codes yet 16:58
So far mine goes up until test case 7 which takes around 94 seconds 16:58
Nikmal Niazi
Has the other two people sent their codes yet
Alana has send it her code on teams ,did @~Abdalla send his code? 17:02
Nikmal Niazi
Don't know 17:02
@~Abdalla He said before he was going to send it but he might have forgotten 17:03
Nikmal Niazi
Yo @Bryan what's your running times 17:07
Nikmal Niazi
Yo @Bryan what's your running times
i dont remember but give me a few minutes, i will run it and test it 17:42
for test 4 it takes 7.154895seconds 18:11
Nikmal Niazi
Yours is a bit slow 18:16
yes, if you one run faster lets use your one and work on it 18:18
Nikmal Niazi
I've already started working on it 18:22
It's on teams 18:22
My original code 18:22
We are using my one and its on teams so if everyone could help by working on it 18:23

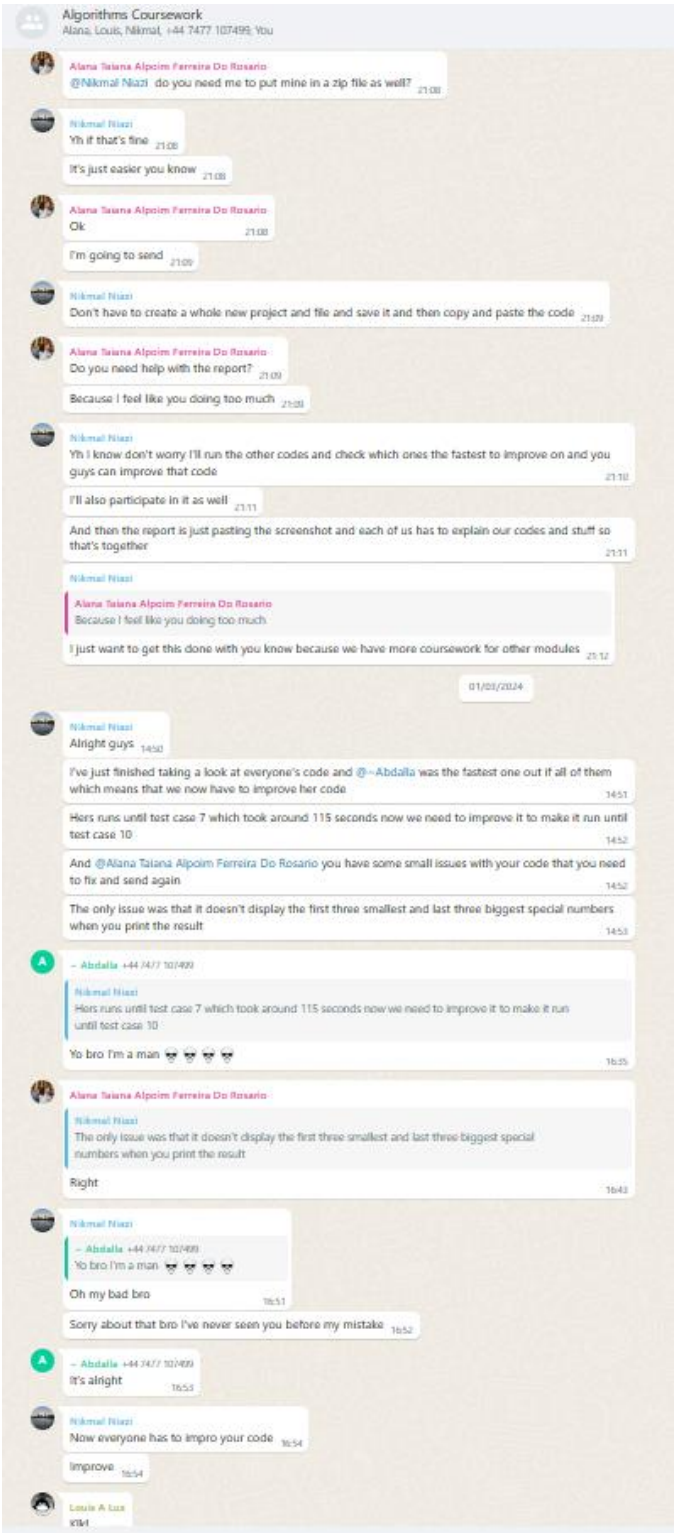
- Figure 11


Algorithms Coursework
Alana, Louis, Nikmal, +44 7477 107499, You
28/02/2024
Nikmal Niazi
Yo @Bryan 12:23
I've finished improving my code 12:24
Got it to about 10 seconds for test case 10 12:24
Nikmal Niazi
12:26
This is the result for the last test case 12:26
Now I'm going to send that over to you on teams and we need one other person's code to improve and we have basically finished 12:27
Just need to do the report as well 12:27
Nikmal Niazi
Got it to about 10 seconds for test case 10
that's great 13:37
Nikmal Niazi
Now I'm going to send that over to you on teams and we need one other person's code to improve and we have basically finished
yeah that fine, send your code and i will start to fill in the report 13:39
Nikmal Niazi
You
yeah that fine, send your code and i will start to fill in the report
I'll send it in a bit, just need making it neat and a bit shorter 13:40
We still need to check who else has the fastest code to improve on 13:40

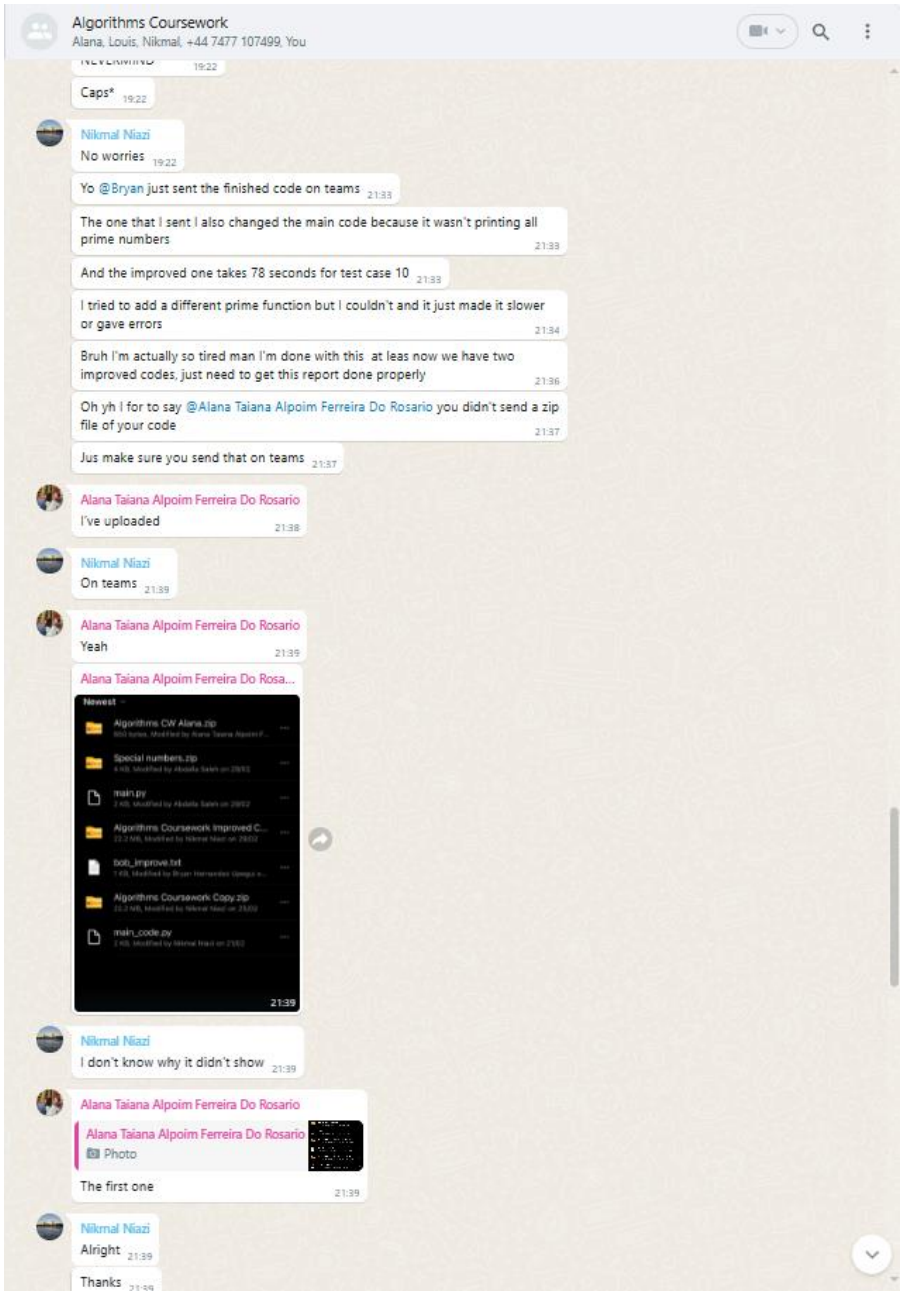
- Figure 12


Nikmal Niazi
Yo @Bryan I've sent over the code now it's very nice and neat 19:59
It took hours of coding for 3 days to get it down that much 19:59
Please don't share the code with anyone 19:59
My one runs 7 seconds for test case 10 20:00
The last one 20:00
We still need one more code tho to improve in order to get onto the report and we still haven't received @~Abdalla code 20:01
Louis A Luz
Are we just gonna put these 2 codes on the report 20:41
Nikmal Niazi
No 20:41
The report you have to put every all of the codes together and explain it with the two improved 20:43
There's like different tasks 20:43
Louis A Luz
Ah Kkk 20:43
~ Abdalla +44 7477 107400
Apologies for the delays guys I have sent my code file and copy paste on teams 20:53
Was struggling with work this I hope you can bear with me 20:54
This week^ 20:54
Nikmal Niazi
We need a zip file of your code so that we can take screenshots of the running times and past it into the word document 20:55
~ Abdalla +44 7477 107400
I see 20:56
Let me export it as a zip file give me a couple mins just on my pc rn 20:56
Nikmal Niazi
Oo 20:56
Ok 20:56
~ Abdalla +44 7477 107400
@Nikmal Niazi I have sent it as a zipped file do you mind checking for me ? 21:04
If it's okay ? 21:04
Nikmal Niazi
Yh it's fine 21:05
~ Abdalla +44 7477 107400
Lovely 21:07
Nikmal Niazi
@Bryan can you check if it works for you 21:08
Alana Taiana Alpoim Ferreira Do Rosario
@Nikmal Niazi do you need me to put mine in a zip file as well? 21:08
Nikmal Niazi
Yh if that's fine 21:08
It's just easier you know 21:08
Alana Taiana Alpoim Ferreira Do Rosario
Ok 21:08
I'm going to send 21:09

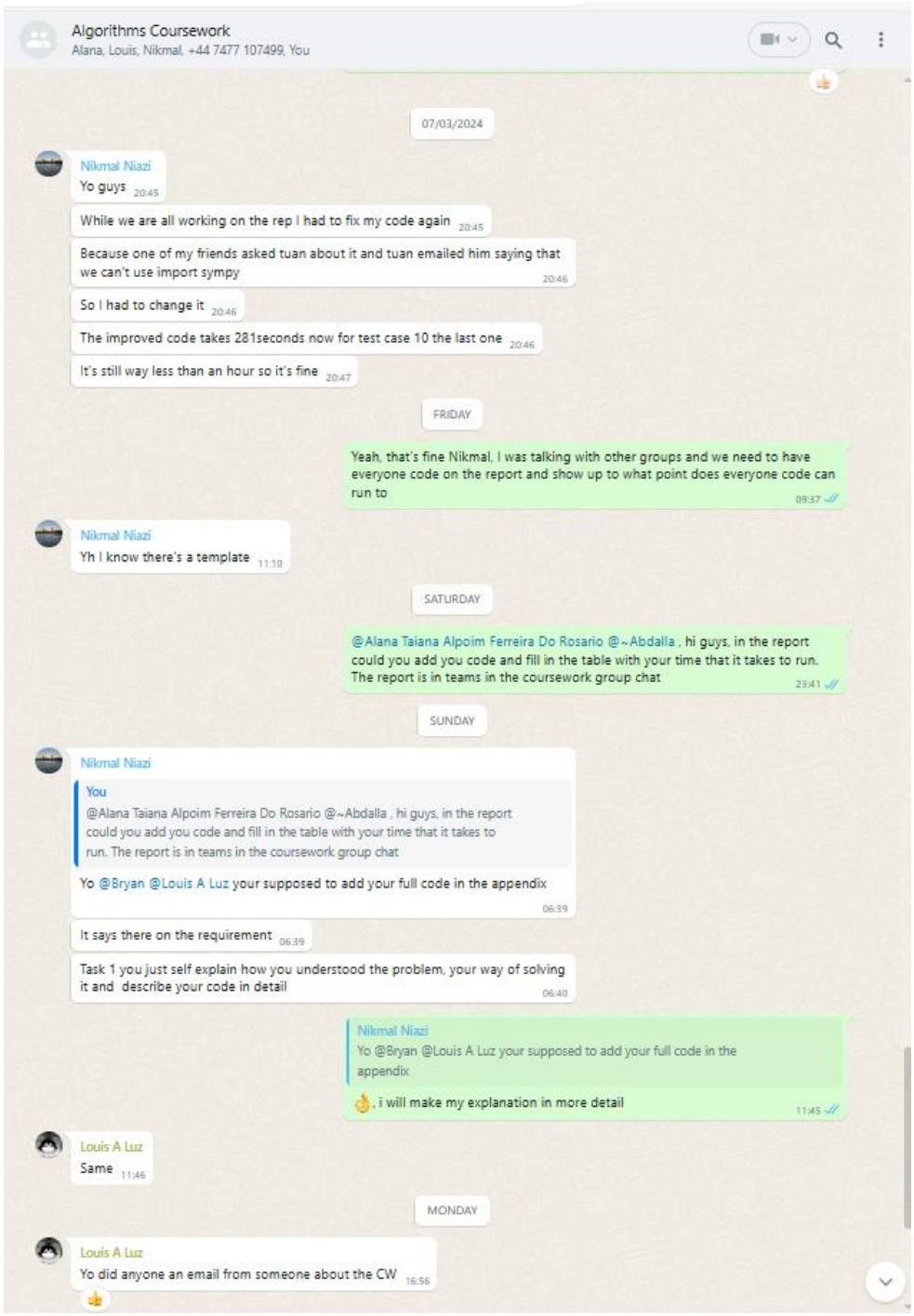
- Figure 13


Algorithms Coursework
Alana, Louis, Nikmal, +44 7477 107499, You
Alana Taiana Alpoim Ferreira Do Rosario
@Nikmal Niazi do you need me to put mine in a zip file as well? 21:08
Nikmal Niazi
Yh if that's fine 21:08
It's just easier you know 21:08
Alana Taiana Alpoim Ferreira Do Rosario
Ok 21:08
I'm going to send 21:09
Nikmal Niazi
Don't have to create a whole new project and file and save it and then copy and paste the code 21:09
Alana Taiana Alpoim Ferreira Do Rosario
Do you need help with the report? 21:09
Because I feel like you doing too much 21:09
Nikmal Niazi
Yh I know don't worry I'll run the other codes and check which ones the fastest to improve on and you guys can improve that code 21:10
I'll also participate in it as well 21:11
And then the report is just pasting the screenshot and each of us has to explain our codes and stuff so that's together 21:11
Nikmal Niazi
Alana Taiana Alpoim Ferreira Do Rosario
Because I feel like you doing too much
I just want to get this done with you know because we have more coursework for other modules 21:12
01/03/2024
Nikmal Niazi
Alright guys 14:50
I've just finished taking a look at everyone's code and @~Abdalla was the fastest one out if all of them which means that we now have to improve her code 14:51
Hers runs until test case 7 which took around 115 seconds now we need to improve it to make it run until test case 10 14:52
And @Alana Taiana Alpoim Ferreira Do Rosario you have some small issues with your code that you need to fix and send again 14:52
The only issue was that it doesn't display the first three smallest and last three biggest special numbers when you print the result 14:53
~ Abdalla +44 7477 107499
Nikmal Niazi
Hers runs until test case 7 which took around 115 seconds now we need to improve it to make it run until test case 10
Yo bro I'm a man 16:35
Alana Taiana Alpoim Ferreira Do Rosario
Nikmal Niazi
The only issue was that it doesn't display the first three smallest and last three biggest special numbers when you print the result
Right 16:43
Nikmal Niazi
~ Abdalla +44 7477 107499
Yo bro I'm a man
Oh my bad bro 16:51
Sorry about that bro I've never seen you before my mistake 16:52
~ Abdalla +44 7477 107499
It's alright 16:53
Nikmal Niazi
Now everyone has to impro your code 16:54
Improve 16:54
Louis A Lux

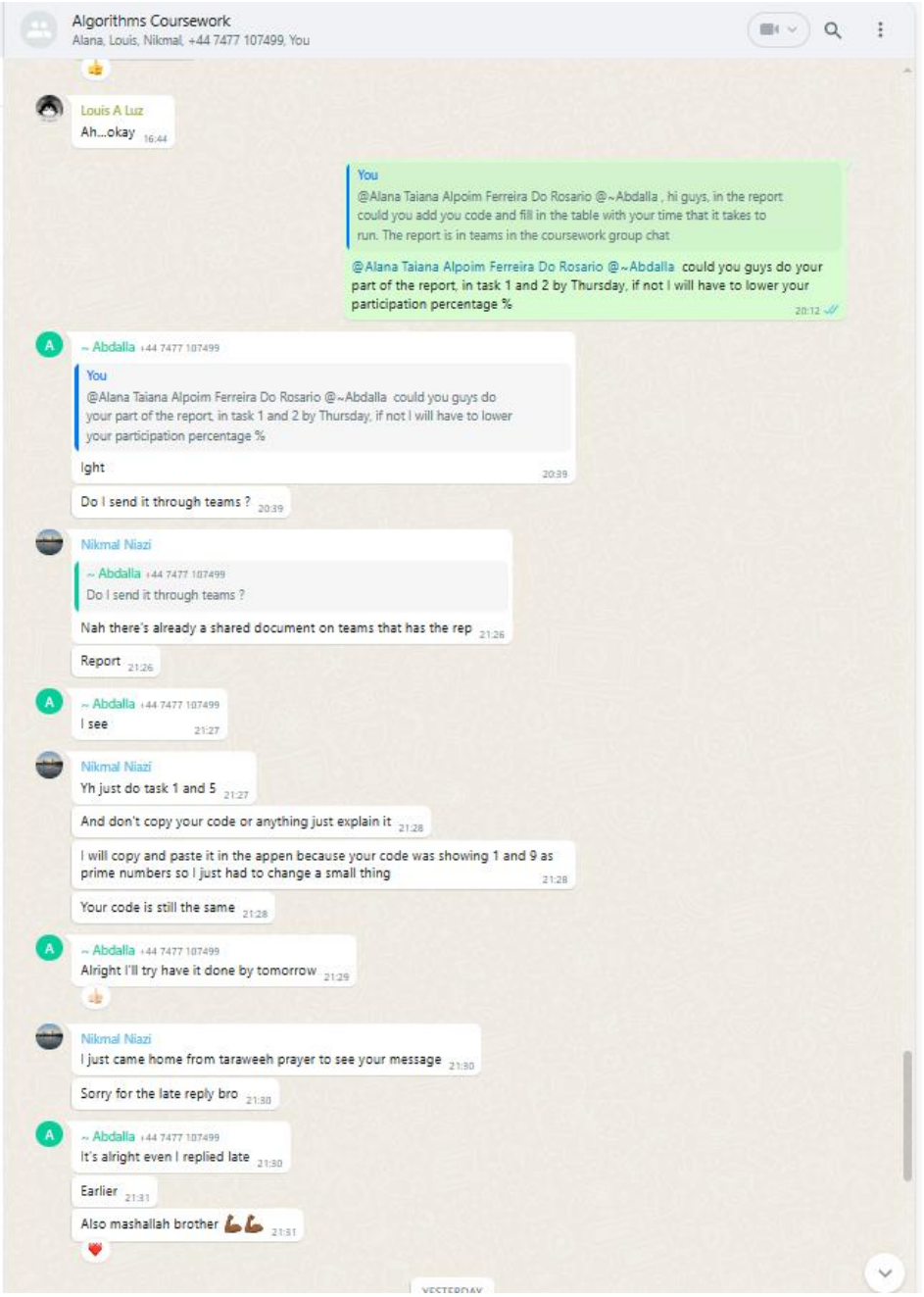
- Figure 14


Algorithms Coursework
Alana, Louis, Nikmal, +44 7477 107499, You
Caps* 19:22
Nikmal Niazi
No worries 19:22
Yo @Bryan just sent the finished code on teams 21:33
The one that I sent I also changed the main code because it wasn't printing all prime numbers 21:33
And the improved one takes 78 seconds for test case 10 21:33
I tried to add a different prime function but I couldn't and it just made it slower or gave errors 21:34
Bruh I'm actually so tired man I'm done with this at leas now we have two improved codes, just need to get this report done properly 21:36
Oh yh I for to say @Alana Taiana Alpoim Ferreira Do Rosario you didn't send a zip file of your code 21:37
Jus make sure you send that on teams 21:37
Alana Taiana Alpoim Ferreira Do Rosario
I've uploaded 21:38
Nikmal Niazi
On teams 21:39
Alana Taiana Alpoim Ferreira Do Rosario
Yeah 21:39
Alana Taiana Alpoim Ferreira Do Rosa...
Newest
Algorithms CW Alana.zip
Special numbers.zip
main.py
Algorithms Coursework Improved C...
bob_improve.txt
Algorithms Coursework Copy.zip
main_code.py
21:39
Nikmal Niazi
I don't know why it didn't show 21:39
Alana Taiana Alpoim Ferreira Do Rosario
Alana Taiana Alpoim Ferreira Do Rosario
Photo
The first one 21:39
Nikmal Niazi
Alright 21:39
Thanks

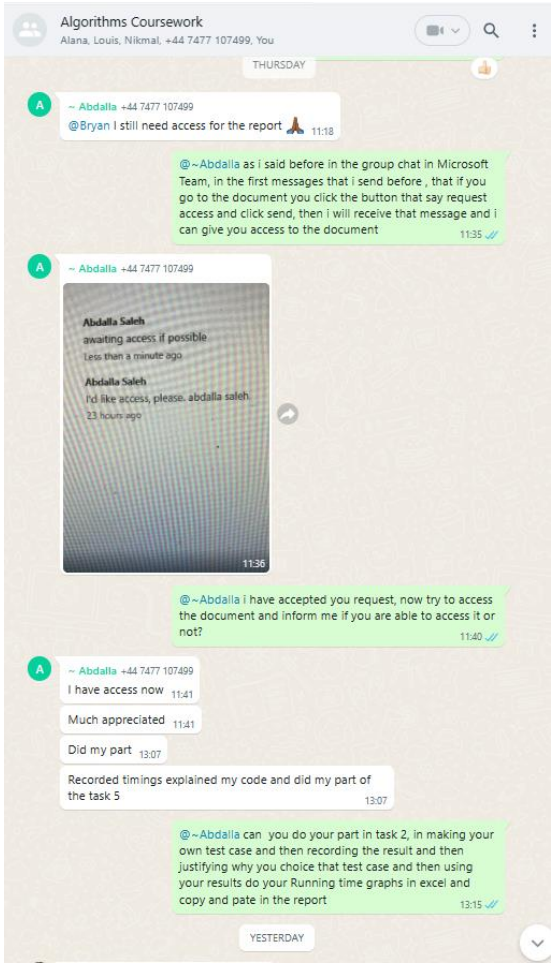
- Figure 15


Algorithms Coursework
Alana, Louis, Nikmal, +44 7477 107499, You
07/03/2024
Nikmal Niazi
Yo guys 20:45
While we are all working on the rep I had to fix my code again 20:45
Because one of my friends asked tuan about it and tuan emailed him saying that we can't use import sympy 20:46
So I had to change it 20:46
The improved code takes 281seconds now for test case 10 the last one 20:46
It's still way less than an hour so it's fine 20:47
FRIDAY
Yeah, that's fine Nikmal, I was talking with other groups and we need to have everyone code on the report and show up to what point does everyone code can run to 09:37
Nikmal Niazi
Yh I know there's a template 11:10
SATURDAY
@Alana Taiana Alpoim Ferreira Do Rosario @~Abdalla , hi guys, in the report could you add you code and fill in the table with your time that it takes to run. The report is in teams in the coursework group chat 23:41
SUNDAY
Nikmal Niazi
You
@Alana Taiana Alpoim Ferreira Do Rosario @~Abdalla , hi guys, in the report could you add you code and fill in the table with your time that it takes to run. The report is in teams in the coursework group chat
Yo @Bryan @Louis A Luz your supposed to add your full code in the appendix 06:39
It says there on the requirement 06:39
Task 1 you just self explain how you understood the problem, your way of solving it and describe your code in detail 06:40
Nikmal Niazi
Yo @Bryan @Louis A Luz your supposed to add your full code in the appendix
, i will make my explanation in more detail 11:45
Louis A Luz
Same 11:46
MONDAY
Louis A Luz
Yo did anyone an email from someone about the CW 16:56

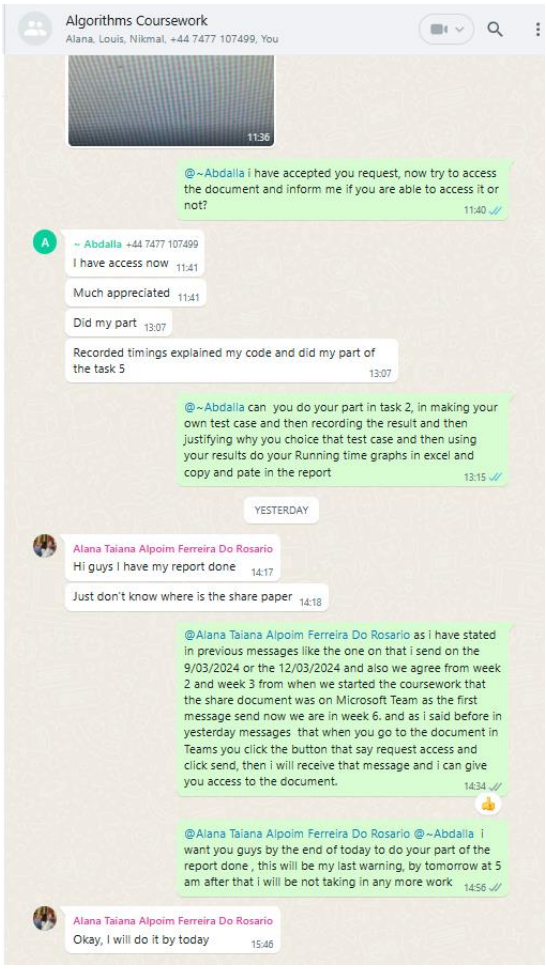
- Figure 16


Algorithms Coursework
Alana, Louis, Nikmal, +44 7477 107499, You
Louis A Luz
Ah...okay 16:44
You
@Alana Taiana Alpoim Ferreira Do Rosario @~Abdalla , hi guys, in the report could you add you code and fill in the table with your time that it takes to run. The report is in teams in the coursework group chat
@Alana Taiana Alpoim Ferreira Do Rosario @~Abdalla could you guys do your part of the report, in task 1 and 2 by Thursday, if not I will have to lower your participation percentage %
20:12
~ Abdalla +44 7477 107499
You
@Alana Taiana Alpoim Ferreira Do Rosario @~Abdalla could you guys do your part of the report, in task 1 and 2 by Thursday, if not I will have to lower your participation percentage %
Ight
20:39
Do I send it through teams ? 20:39
Nikmal Niazi
~ Abdalla +44 7477 107499
Do I send it through teams ?
Nah there's already a shared document on teams that has the rep 21:26
Report 21:26
~ Abdalla +44 7477 107499
I see 21:27
Nikmal Niazi
Yh just do task 1 and 5 21:27
And don't copy your code or anything just explain it 21:28
I will copy and paste it in the appen because your code was showing 1 and 9 as prime numbers so I just had to change a small thing 21:28
Your code is still the same 21:28
~ Abdalla +44 7477 107499
Alright I'll try have it done by tomorrow 21:29
Nikmal Niazi
I just came home from taraweeh prayer to see your message 21:30
Sorry for the late reply bro 21:30
~ Abdalla +44 7477 107499
It's alright even I replied late 21:30
Earlier 21:31
Also mashallah brother 21:31

- Figure 17

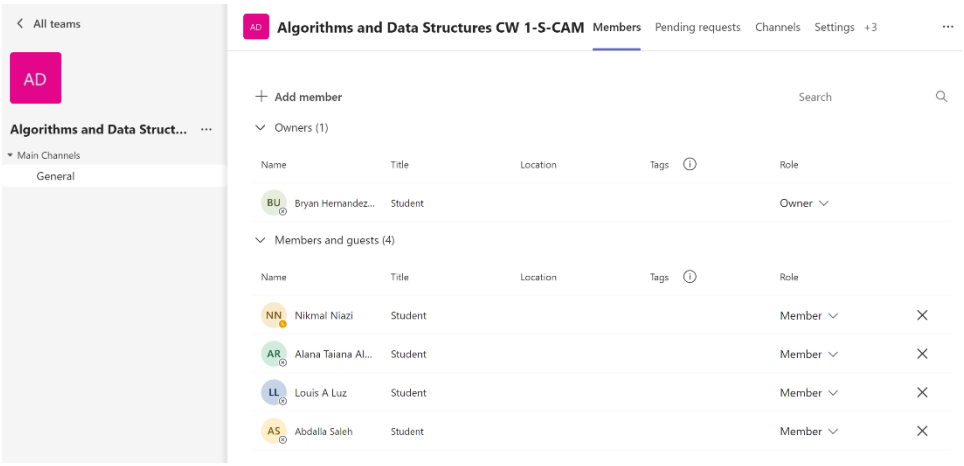

Algorithms Coursework
Alana, Louis, Nikmal, +44 7477 107499, You
THURSDAY
~ Abdalla +44 7477 107499
@Bryan I still need access for the report 11:18
@~Abdalla as i said before in the group chat in Microsoft Team, in the first messages that i send before , that if you go to the document you click the button that say request access and click send, then i will receive that message and i can give you access to the document 11:35
~ Abdalla +44 7477 107499
Abdalla Saleh
awaiting access if possible
Less than a minute ago
Abdalla Saleh
I'd like access, please. abdalla saleh
23 hours ago
11:36
@~Abdalla i have accepted you request, now try to access the document and inform me if you are able to access it or not? 11:40
~ Abdalla +44 7477 107499
I have access now 11:41
Much appreciated 11:41
Did my part 13:07
Recorded timings explained my code and did my part of the task 5 13:07
@~Abdalla can you do your part in task 2, in making your own test case and then recording the result and then justifying why you choice that test case and then using your results do your Running time graphs in excel and copy and pate in the report 13:15
YESTERDAY

- Figure 18

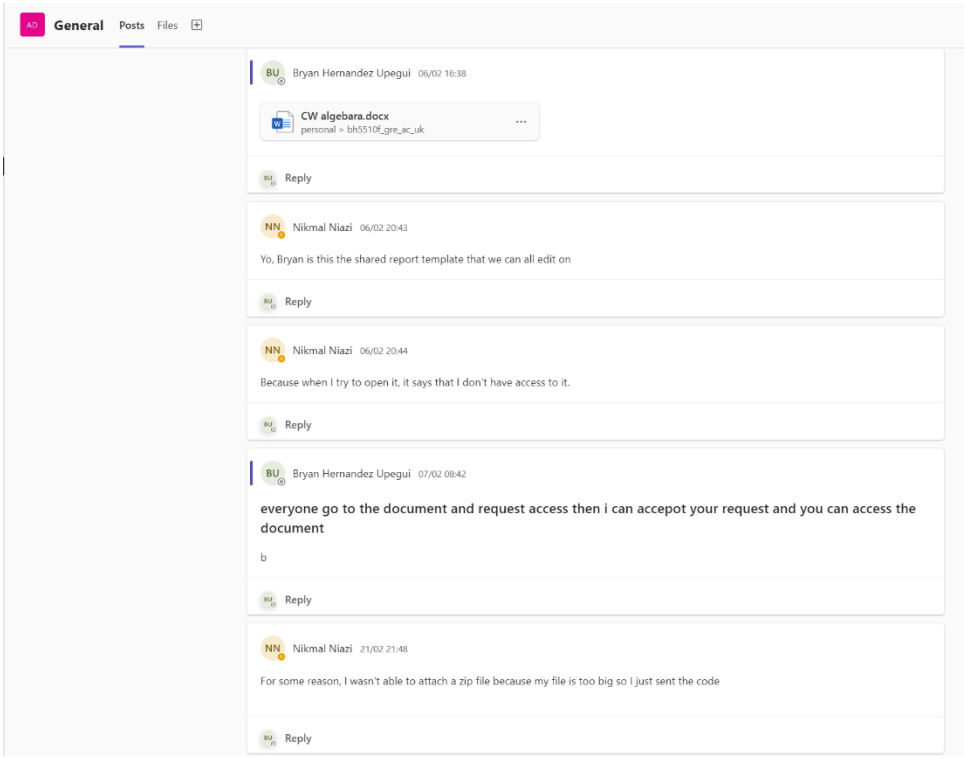


Communication logs – Microsoft Teams (group chat, mainly used to send code and files)

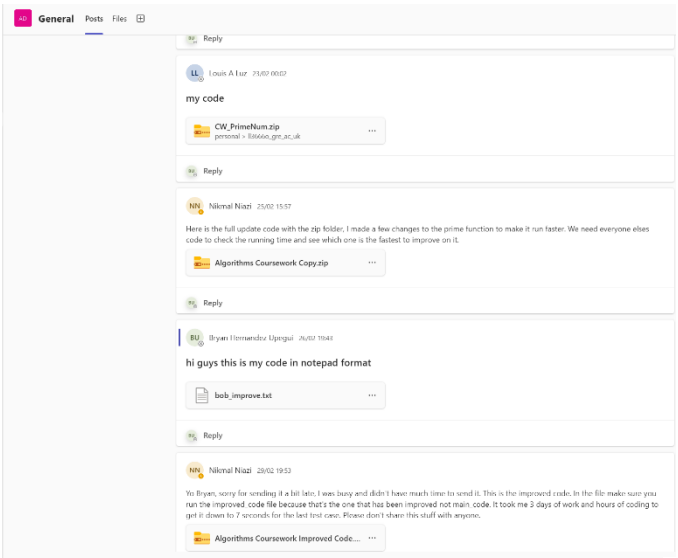
This is a screenshot of the group chat in Teams, it was made by Bryan the group leader and here are all the members that are in the group.

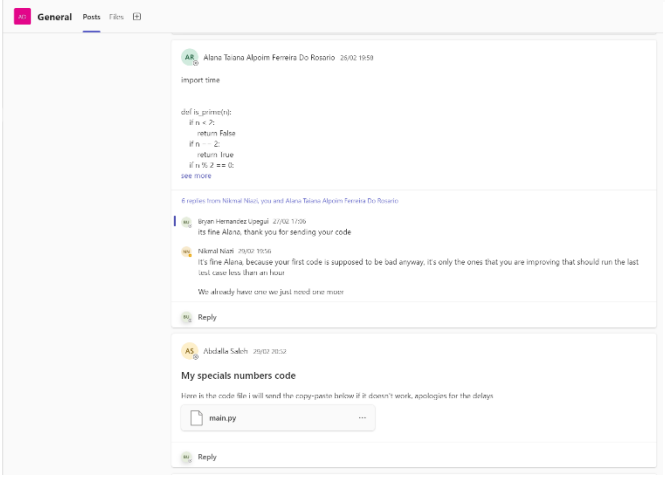


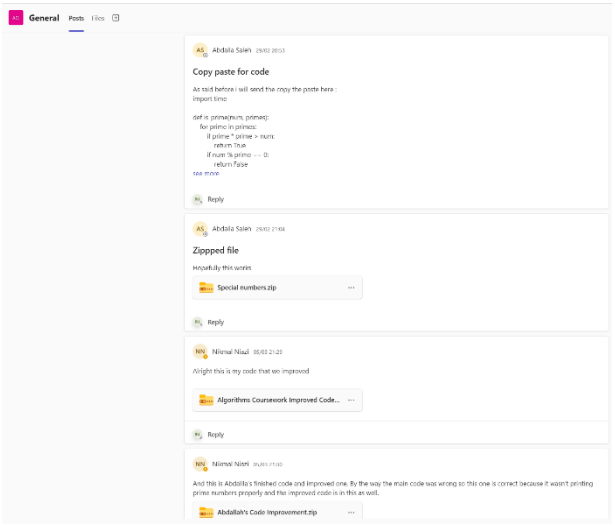
These are the first messages, here you can see the group leader Bryan shared the report that we can all the teams members can work on and for team's member to get access for the first time they must request access for the document after they send a request, they can have access to the document.

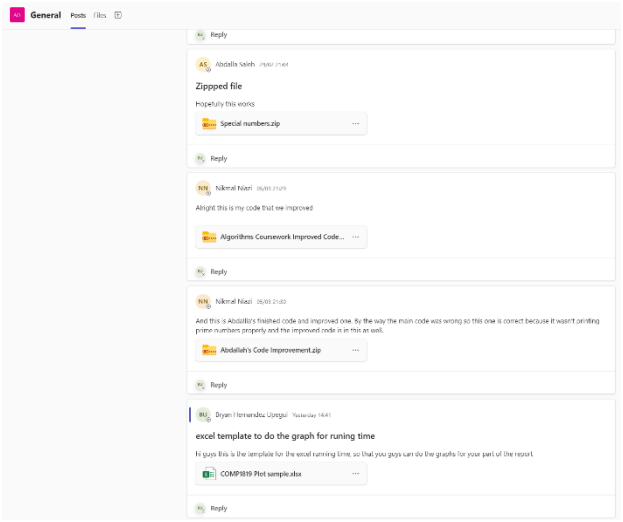


For the next screenshots Teams was used to share documents, files and code. WhatsApp was used mainly to communicate.


General
Posts
Files
Reply
Louis A Luz 25/02 00:07
my code
CW_PrimeNum.zip
personal > ll66466_gre_ac_uk
Reply
Nikmal Niazi 25/02 15:57
Here is the full update code with the zip folder, I made a few changes to the prime function to make it run faster. We need everyone elses code to check the running time and see which one is the fastest to improve on it.
Algorithms Coursework Copy.zip
Reply
Bryan Hernandez Upegui 26/02 19:48
hi guys this is my code in notepad format
bob_improve.txt
Reply
Nikmal Niazi 28/02 19:53
Yo Bryan, sorry for sending it a bit late, I was busy and didn't have much time to send it. This is the improved code. In the file make sure you run the improved_code file because that's the one that has been improved not main_code. It took me 3 days of work and hours of coding to get it down to 7 seconds for the last test case. Please don't share this stuff with anyone.
Algorithms Coursework Improved Code....


General
Posts
Files
Alana Taiana Alpoim Ferreira Do Rosario 26/02 19:59
import time
def is_prime(n):
if n < 2:
return False
if n == 2:
return True
if n % 2 == 0:
see more
6 replies from Nikmal Niazi, you and Alana Taiana Alpoim Ferreira Do Rosario
Bryan Hernandez Upegui 27/02 7:09
its fine Alana, thank you for sending your code
Nikmal Niazi 28/02 19:56
It's fine Alana, because your first code is supposed to be bad anyway, it's only the ones that you are improving that should run the last test case less than an hour
We already have one we just need one more
Reply
Abdalla Saleh 28/02 20:52
My specials numbers code
Here is the code file I will send the copy-paste below if it doesn't work, apologies for the delays
main.py
Reply


General
Posts
Files
Abdalla Saleh 28/02 20:53
Copy paste for code
As said before I will send the copy the paste here :
import time
def is_prime(num, primes):
for prime in primes:
if prime * prime > num:
return True
if num % prime == 0:
return False
see more
Reply
Abdalla Saleh 28/02 21:04
Zipped file
Hopefully this works
Special numbers.zip
Reply
Nikmal Niazi 28/02 21:23
Alright this is my code that we improved
Algorithms Coursework Improved Code...
Reply
Nikmal Niazi 01/03 21:10
And this is Abdalla's finished code and improved one. By the way the main code was wrong so this one is correct because it wasn't printing prime numbers properly and the improved code is in this as well.
Abdallah's Code Improvement.zip



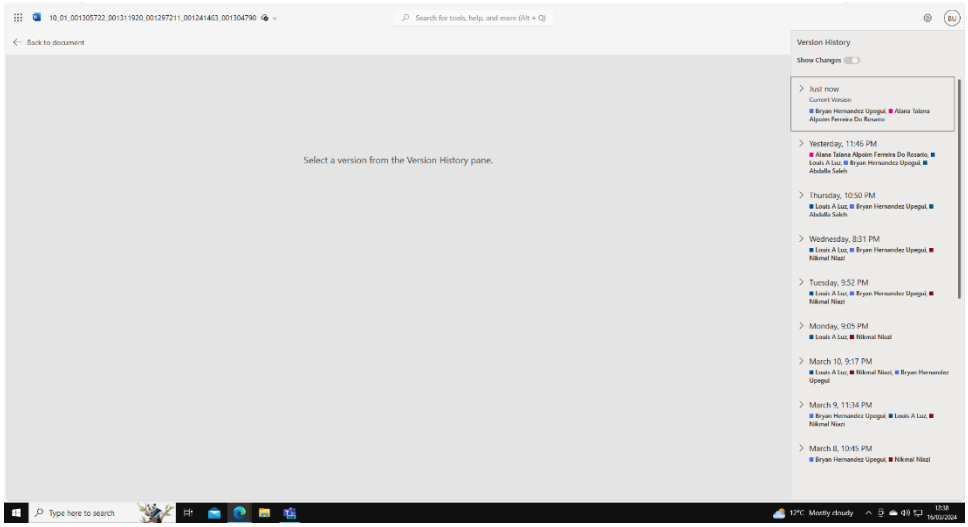
<u>Communication logs – Microsoft word</u>

(we used word to communicate by adding comment in section when wanted to ask someone something or tell information for a section in the report)

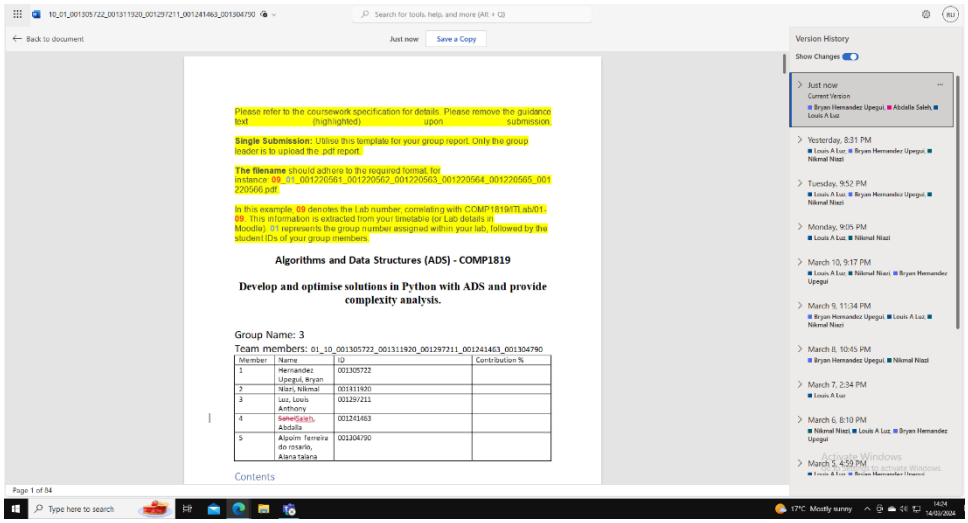
These are screenshots of the history of the document to show who has been working on the report.

As seen on picture 1 and 2, the main people who have been working on the report are Bryan the group leader, Nikmal and Louis, from the 6 of February to the 13 of march, on the 14march Abdalla first join to do some of his part of the report as seen on picture 3, in the 15 of march Friday as seen in picture 4 Alana for the first time in all the time working on the coursework join to do her part of the coursework.

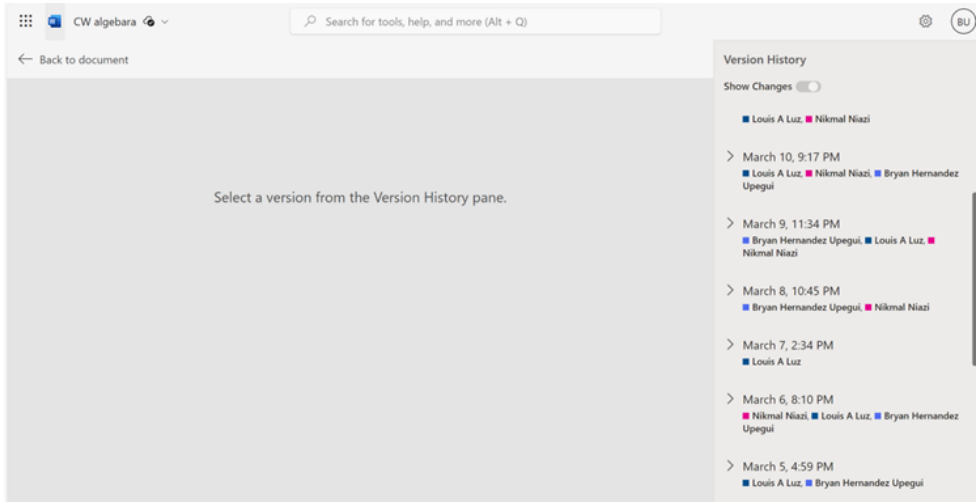
- Picture 4



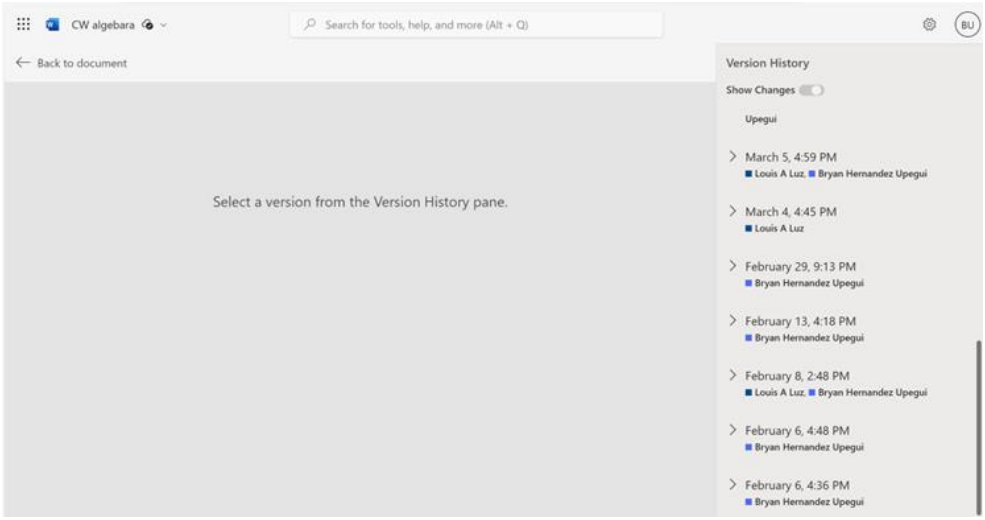
- Picture 3



- Picture 2



- Picture 1



- This screenshot is an example how we communicated in word, using comments, to tell teams mates what to do for some parts.