HyperionDev Take Home Test Solution: Coding Mentor Code Review.

By: OluwaKemmy Mary O Jones

Section A

Recommended Universal IDE: Microsoft Visual Studio Code.
Download link: Download Visual Studio Code[i]

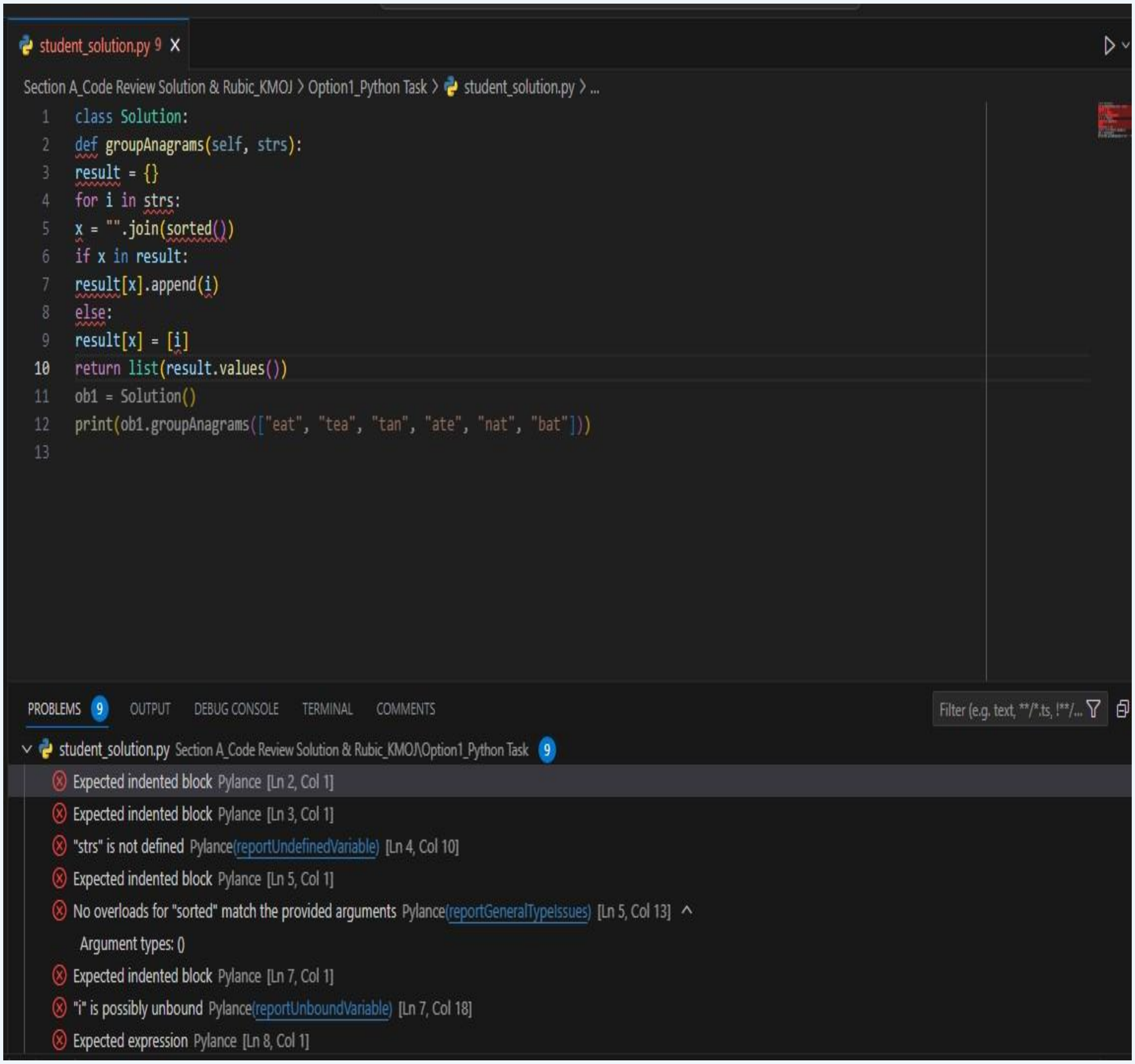
Option 1: Python Task (anagram.py)						
CODE REVIEW RUBIC	**GRADE & MARKS** **10 Marks Each**			**CODE LINE**	**ERRORS / ISSUES TO RESOLVE**	**MENTOR'S REMARK**
Code Correctness:	~Correct file name: ~ Displays Output: ~ Clean Coding: Total Marks (30): Grade:	Yes No Yes => =>	10 0 10 20 C+	2-12	Error due to wrong & irregular indentation resulting to the output being unreachable.	~Fix the indentation error by setting the indentation in the code correctly. ~No comments, hence understanding what each line or chunk of code does may be unclear and can make code maintenance difficult in the future, in case it is assigned to a new programmer of less experience (e.g., a beginner).
Efficiency:	~Code Reliability: ~ Speed to compile: ~ Methodology: Total Marks (30): Grade:	Fair Excl. Good => =>	6 10 8 24 A	2-10	~Code may be prone to error, due to lack of explicit definition of some variable even though generic coding is used. ~Code output is hard-coded, lack of user input. ~No entry point, i.e., main() method.	~It is good practice to always have a main () method which is an entry point and display the output of your program resulting in a neater, and the input codes in modules such as functions, methods this helps enhance the code and result in a neater organized reusable code. ~It is good practice to let your program accept user input, which is then processed, and the compiler displays the desired output it enhances user experience. ~It is good practice to use functions and methods to compartmentalize code thus modularizing codes enhance readability of code syntax and aids easy debugging of code in case of any error during code compile time or run time.
Style:	~Indentation: ~Alignment: ~Name Convention: Total Marks (30): Grade:	No No Fair => =>	0 0 7 7 F	1,2,4	~Code is not properly Indented nor aligned. ~Class name is not unique. ~Variable x's name is not meaningful since it is a string or unique. ~Function name in python is preferred to be written in lower-case and joined with hyphen.[ii]	~It is good practice to let Class name match the file name. It helps you or any programmer assigned to maintain the code to be able to identify which class is called your code contains more than one class. ~Name the function *groupAnagrams preferably as group_anagrams.* *See: Python naming convention for functions.*
Documentation	~Comments: ~Documentation: Total Marks (10): Grade:	No No => =>	0 0 0 F	1-12	~Lack of Comments & code documentation is not good practice and may make reading & understanding what the code does and how to maintain or fix errors may be difficult.	~It is good practice to include comments and documentation in a program. ~It enhances the code to be more comprehensive, and easier to understand what each code does and makes code debugging, improvement easier.
Possible Improvement to Enhance Code	-Fix and clear indentation error and other errors. -Make code more readable and maintainable because by adding comments in your code on what major codes lines do, it is very important as it enhances code readability and code maintainability. -It is good practice to always add an entry point (main () method) to your code, to make it easier for your compiler to run and compile your code. -It is good practice to follow python language naming convention and indentations rules. -It is very good practice to create a program that takes input from users, easy to use by users and leaving user instructions on how to proceed when using the program. -It is good practice to add error handling and input validation. **References**[iii]					

Student's anagram.py syntax

This code has errors because:

1. There was no proper indentation and

2. The lines of codes are not aligned according to python convention.



student_solution.py 9 X

Section A_Code Review Solution & Rubic_KMOJ > Option1_Python Task > student_solution.py > ...

```python
class Solution:
def groupAnagrams(self, strs):
result = {}
for i in strs:
x = "".join(sorted())
if x in result:
result[x].append(i)
else:
result[x] = [i]
return list(result.values())
ob1 = Solution()
print(ob1.groupAnagrams(["eat", "tea", "tan", "ate", "nat", "bat"]))
```

PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

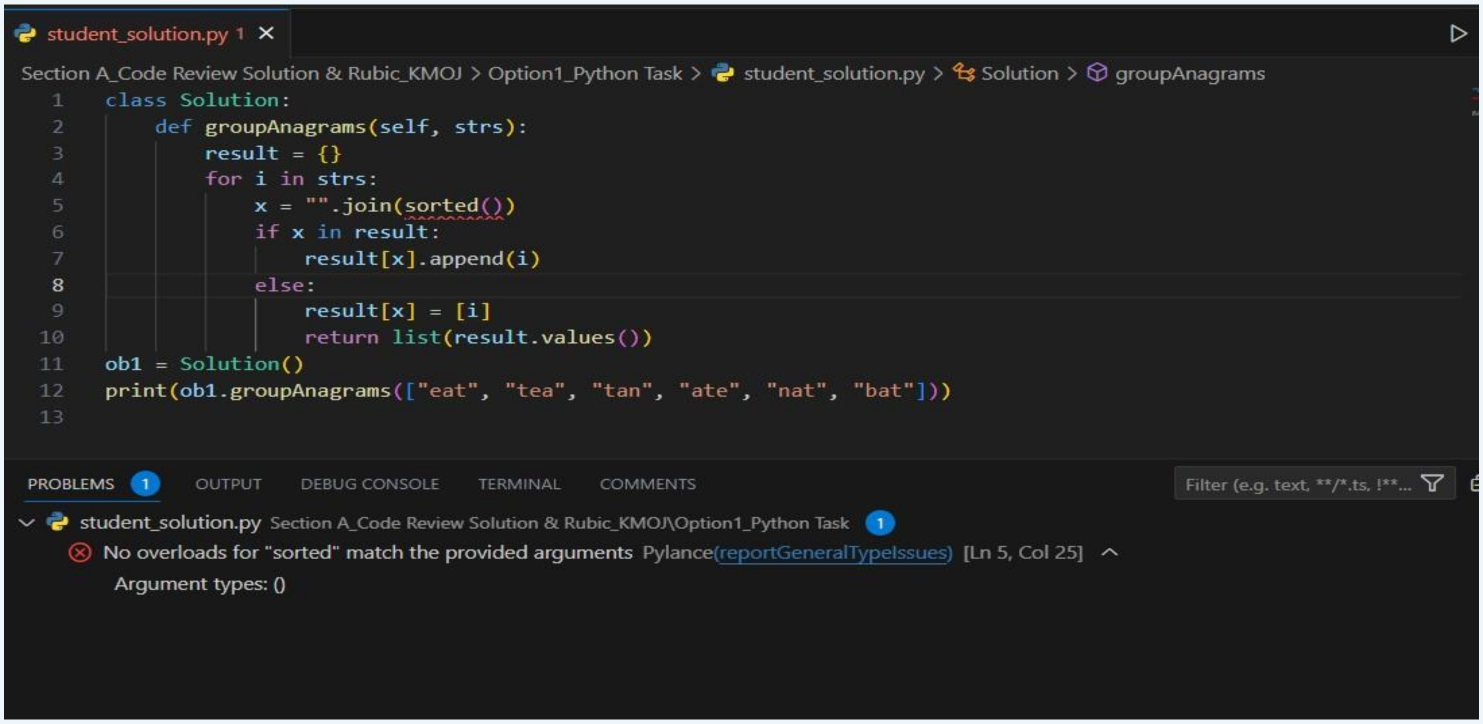
```
student_solution.py Section A_Code Review Solution & Rubic_KMOJ\Option1_Python Task 9
  Expected indented block Pylance [Ln 2, Col 1]
  Expected indented block Pylance [Ln 3, Col 1]
  "strs" is not defined Pylance(reportUndefinedVariable) [Ln 4, Col 10]
  Expected indented block Pylance [Ln 5, Col 1]
  No overloads for "sorted" match the provided arguments Pylance(reportGeneralTypeIssues) [Ln 5, Col 13]
    Argument types: ()
  Expected indented block Pylance [Ln 7, Col 1]
  "i" is possibly unbound Pylance(reportUnboundVariable) [Ln 7, Col 18]
  Expected expression Pylance [Ln 8, Col 1]
```

Steps on how to fix the error in the student's anagram.py file:

1. Clear the indentation error and align the code accordingly.

2. Then clear the type error raised on the Sorted function.

3. Run the code to the output.



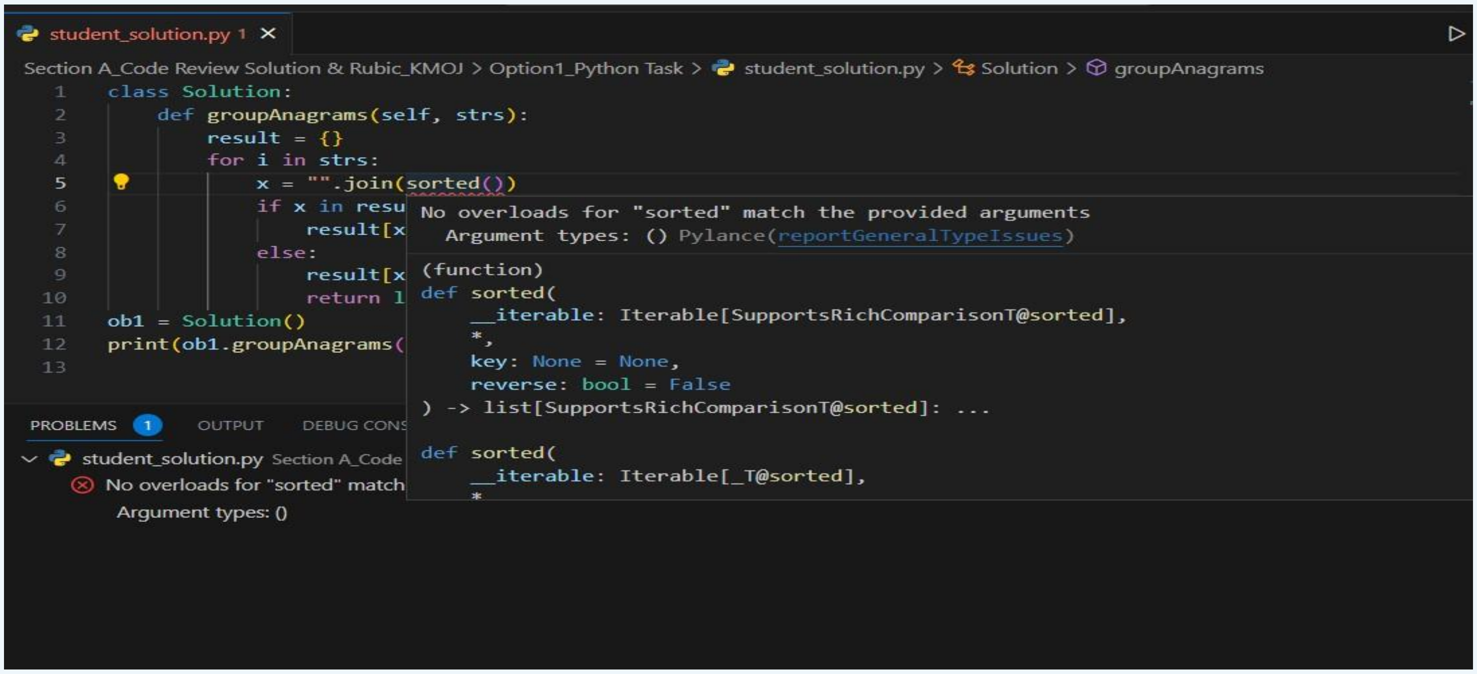
student_solution.py 1

Section A_Code Review Solution & Rubic_KMOJ > Option1_Python Task > student_solution.py > Solution > groupAnagrams

```python
class Solution:
    def groupAnagrams(self, strs):
        result = {}
        for i in strs:
            x = "".join(sorted())
            if x in result:
                result[x].append(i)
            else:
                result[x] = [i]
                return list(result.values())
ob1 = Solution()
print(ob1.groupAnagrams(["eat", "tea", "tan", "ate", "nat", "bat"]))
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
student_solution.py Section A_Code Review Solution & Rubic_KMOJ\Option1_Python Task 1
  No overloads for "sorted" match the provided arguments Pylance(reportGeneralTypeIssues) [Ln 5, Col 25]
    Argument types: ()
```



student_solution.py 1

Section A_Code Review Solution & Rubic_KMOJ > Option1_Python Task > student_solution.py > Solution > groupAnagrams

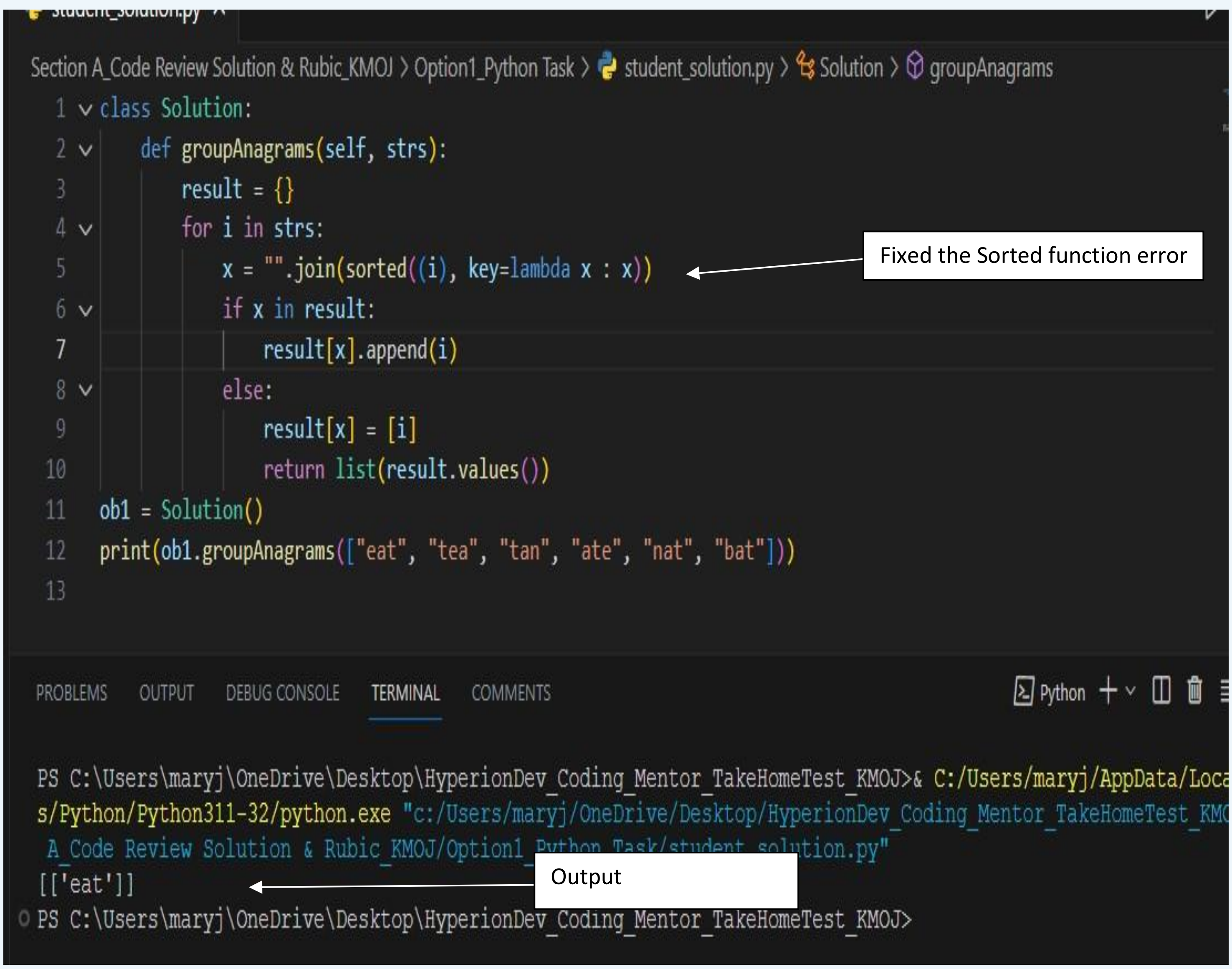
```
No overloads for "sorted" match the provided arguments
  Argument types: () Pylance(reportGeneralTypeIssues)
(function)
def sorted(
    __iterable: Iterable[SupportsRichComparisonT@sorted],
    *,
    key: None = None,
    reverse: bool = False
) -> list[SupportsRichComparisonT@sorted]: ...

def sorted(
    __iterable: Iterable[_T@sorted],
    *
```

Fixing the Sorted() function error in the Student's anagram.py file:

1. The Sorted() function needs to have an argument and a key because it is a function in the ‘groupby’ library in python, hence in order to use the Sorted() function, one needs to iterate through the elements in the list

2. To fix [‘str is not defined or i’] is unbound in the argument and create a lambda function of x implicitly as the key

3. Run the code to the output.


student_solution.py
Section A_Code Review Solution & Rubic_KMOJ > Option1_Python Task > student_solution.py > Solution > groupAnagrams
Fixed the Sorted function error
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS
Python
Output

```python
class Solution:
    def groupAnagrams(self, strs):
        result = {}
        for i in strs:
            x = "".join(sorted((i), key=lambda x : x))
            if x in result:
                result[x].append(i)
            else:
                result[x] = [i]
                return list(result.values())
ob1 = Solution()
print(ob1.groupAnagrams(["eat", "tea", "tan", "ate", "nat", "bat"]))
```

```
PS C:\Users\maryj\OneDrive\Desktop\HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ>& C:/Users/maryj/AppData/Loca
s/Python/Python311-32/python.exe "c:/Users/maryj/OneDrive/Desktop/HyperionDev_Coding_Mentor_TakeHomeTest_KMO
 A_Code Review Solution & Rubic_KMOJ/Option1_Python Task/student_solution.py"
[['eat']]
PS C:\Users\maryj\OneDrive\Desktop\HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ>
```

MENTOR's EXAMPLE SOLUTION (INPUT)

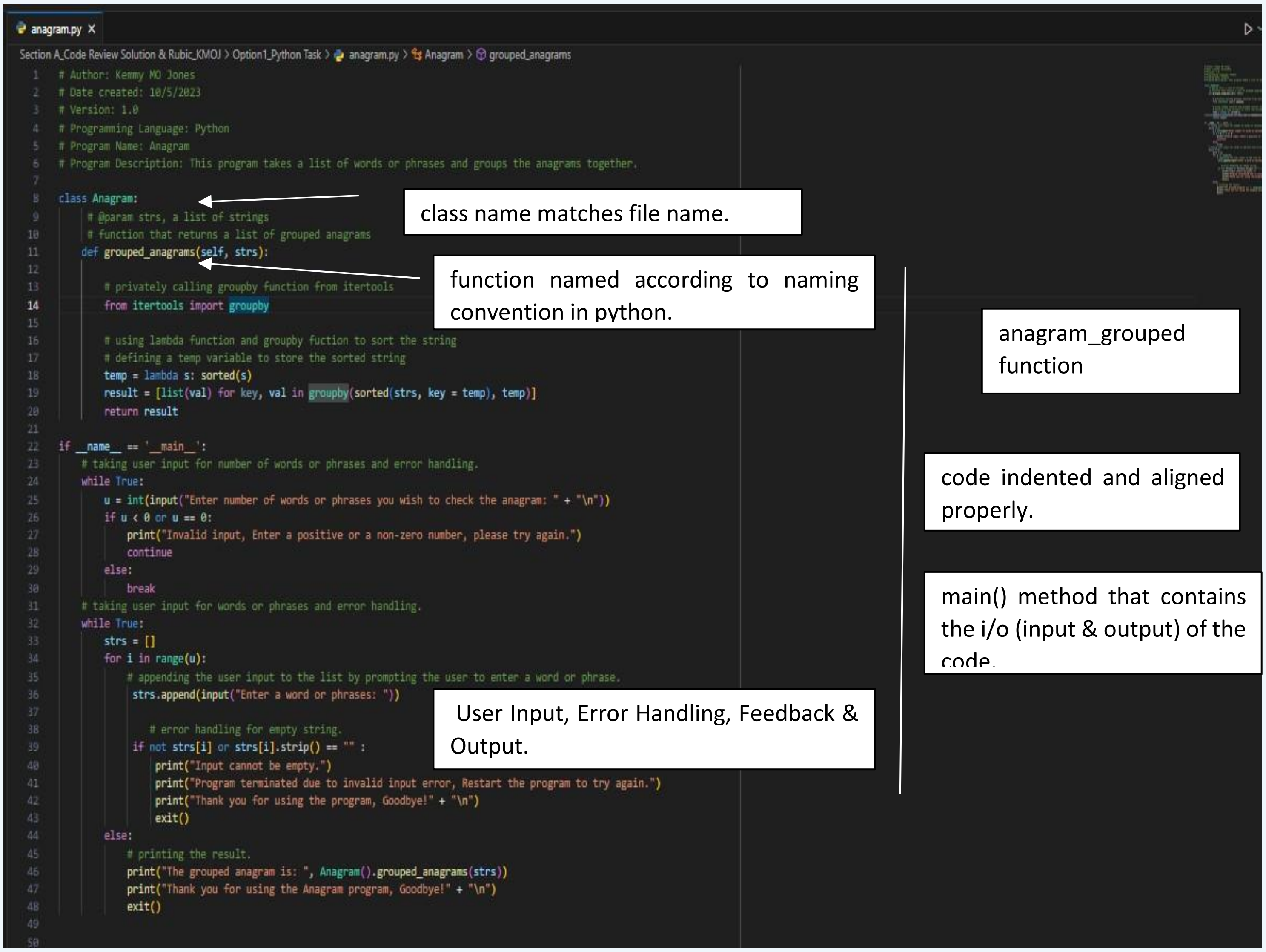
anagram.py file: improvements students can make on their program

1. Prompt user for input and displays the anagram of the input string.

2. Code is compartmentalized and modularized, has main method as the entry point, code validation and error handling for example imputing empty string, etc.

3. Add documentation and comments on important lines or chunk of code describing what the code does.

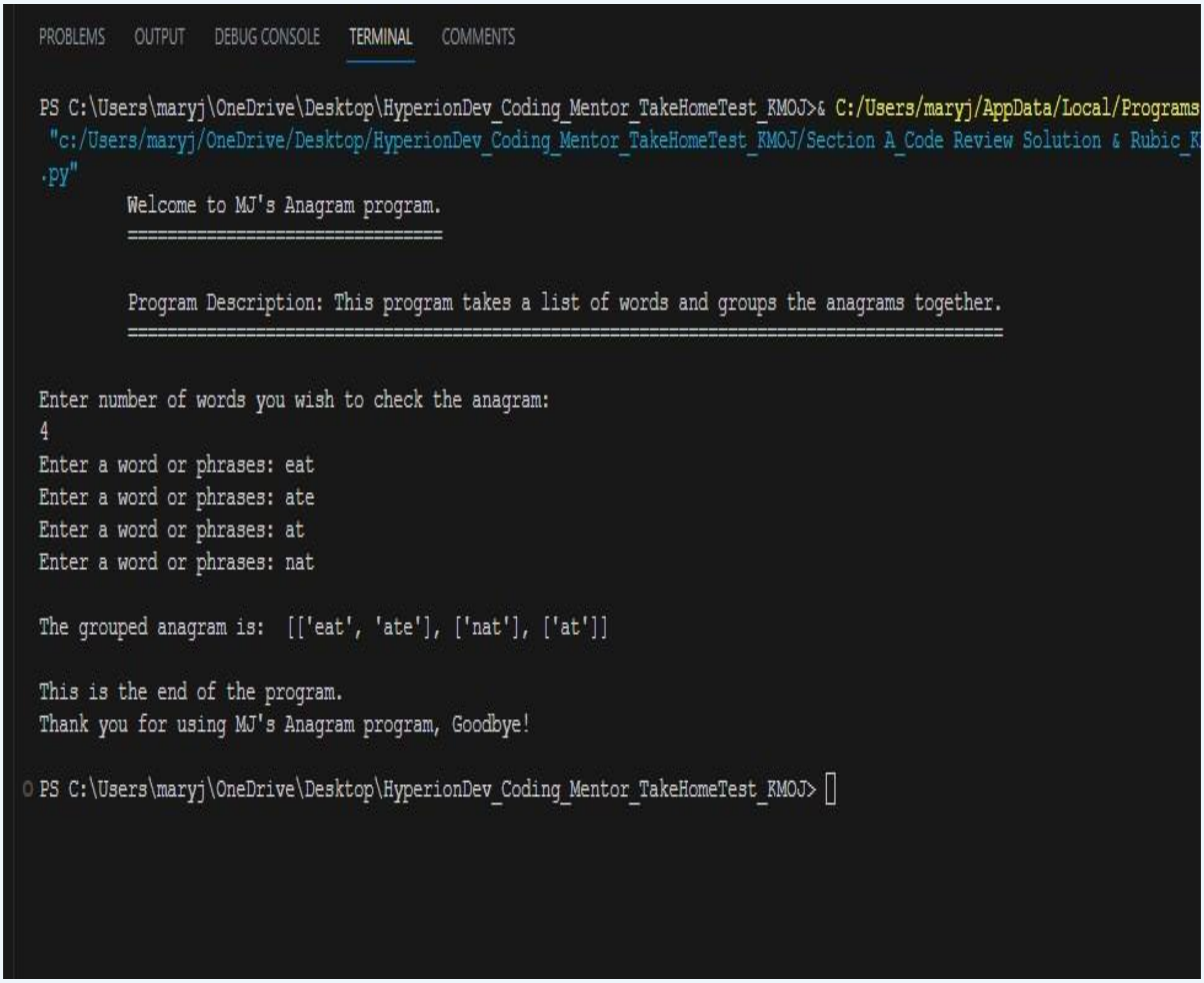
4. Add instruction and feedback to user on the task success or completion and error message display in order to create a better user experience.


anagram.py
Section A_Code Review Solution & Rubic_KMOJ > Option1_Python Task > anagram.py > Anagram > grouped_anagrams
class name matches file name.
function named according to naming convention in python.
anagram_grouped function
code indented and aligned properly.
main() method that contains the i/o (input & output) of the code.
User Input, Error Handling, Feedback & Output.

```python
# Author: Kemmy MO Jones
# Date created: 10/5/2023
# Version: 1.0
# Programming Language: Python
# Program Name: Anagram
# Program Description: This program takes a list of words or phrases and groups the anagrams together.

class Anagram:
    # @param strs, a list of strings
    # function that returns a list of grouped anagrams
    def grouped_anagrams(self, strs):

        # privately calling groupby function from itertools
        from itertools import groupby

        # using lambda function and groupby fuction to sort the string
        # defining a temp variable to store the sorted string
        temp = lambda s: sorted(s)
        result = [list(val) for key, val in groupby(sorted(strs, key = temp), temp)]
        return result

if __name__ == '__main__':
    # taking user input for number of words or phrases and error handling.
    while True:
        u = int(input("Enter number of words or phrases you wish to check the anagram: " + "\n"))
        if u < 0 or u == 0:
            print("Invalid input, Enter a positive or a non-zero number, please try again.")
            continue
        else:
            break
    # taking user input for words or phrases and error handling.
    while True:
        strs = []
        for i in range(u):
            # appending the user input to the list by prompting the user to enter a word or phrase.
             strs.append(input("Enter a word or phrases: "))

                # error handling for empty string.
             if not strs[i] or strs[i].strip() == "" :
                 print("Input cannot be empty.")
                 print("Program terminated due to invalid input error, Restart the program to try again.")
                 print("Thank you for using the program, Goodbye!" + "\n")
                 exit()
        else:
            # printing the result.
            print("The grouped anagram is: ", Anagram().grouped_anagrams(strs))
            print("Thank you for using the Anagram program, Goodbye!" + "\n")
            exit()
```

MENTOR’s EXAMPLE SOLUTION (OUTPUT)

anagram.py file: The anagram program output is displayed as shown below:



```
PS C:\Users\maryj\OneDrive\Desktop\HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ>& C:/Users/maryj/AppData/Local/Programs
 "c:/Users/maryj/OneDrive/Desktop/HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ/Section A_Code Review Solution & Rubic_K
.py"
        Welcome to MJ's Anagram program.
        ================================

        Program Description: This program takes a list of words and groups the anagrams together.
        ==========================================================================================

Enter number of words you wish to check the anagram:
4
Enter a word or phrases: eat
Enter a word or phrases: ate
Enter a word or phrases: at
Enter a word or phrases: nat

The grouped anagram is:  [['eat', 'ate'], ['nat'], ['at']]

This is the end of the program.
Thank you for using MJ's Anagram program, Goodbye!

PS C:\Users\maryj\OneDrive\Desktop\HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ> 
```

Option 2: Java Task (recursion.java)						
CODE REVIEW RUBIC	GRADE & MARKS 10 Marks Each			CODE LINE	ERRORS / ISSUES TO RESOLVE	MENTOR'S REMARK
Code Correctness:	~Correct file name: ~ Displays Output: ~ Clean Coding: Total Marks (30): Grade:	Yes No Yes => =>	10 0 8 18 C	1-38 19, 26, 28	Errors due to: ~Unclosed Literal String ~Duplicate Type error raised after closing all opened curly braces due to incorrect use of generic coding.	~Fix and clear all errors raised because of unassigned or wrongfully assigned variables. ~Ensure all opened curly braces are closed to avoid memory leak and EOF (end of file) reached error. ~ Fix any wrong concatenation of strings (Literal Strings) to Int or any other type of variable
Efficiency:	~Code Reliability: ~ Speed to compile: ~ Methodology: Total Marks (30): Grade:	Fair Excl. Good => =>	6 10 6 22 B	7,17-27	~Code may be prone to error, due to lack of explicit definition of some variable even though generic coding is used. ~Code output is hard-coded, lack of user input. ~ Literal Strigs is not handled properly.	~It is good practice to always have a main () method which is an entry point and display the output of your program resulting in a neater, and the input codes in modules such as functions, methods this helps enhance the code and result in a neater organized reusable code. ~It is good practice to let your program accept user input which is then processed, and the compiler displays the desired output it enhances user experience. ~It is good practice to use functions and methods to compartmentalize code thus modularizing codes enhance readability of code syntax and aids easy debugging of code in case of any error during code compile time or run time.
Style:	~Indentation: ~Alignment: ~Name Convention: Total Marks (30): Grade:	Yes Yes Excl => =>	10 10 10 30 A	1-38	To Learn more, read this: Java Tutorials.[iv]	~It is good practice to let Class name match the file name. It helps you or any programmer assigned to maintain the codes to be able to identify which class is called in case your code contains more than one class. ~Be extra careful when using generic codes and implicitly defining variables may cause unwanted code leak, compile error or runtime errors
Documentation	~Comments: ~Documentation: Total Marks (10): Grade:	Yes Yes => =>	5 3 8 A	3,5,916, 21, 28 - 30	~Documentation can be improved.	~It is good practice to include comments and documentation in a program it enhances the code to be more comprehensive, easy to understand what each code does and makes code debugging, improvement easier.
Possible Improvement to Enhance Code	-Fix and clear Literal strings concatenation errors. -Make code more readable and maintainable because by adding comments in your code on what major codes lines do, it is very important as it enhances code readability and code maintainability. -It is good practice to break your codes into simple functional set of code chunks or modularized with methods and functions, makes it neater, readable and helps make debugging easy. -Ensure to call necessary parameters, explicit definition of variables and careful use of generic coding when necessary but do not use always. -It is very good practice to create a program that takes input from users, easy to use by users and leaving user instructions on how to proceed when using the program. -It is good practice to add error handling and input validation. References[v]					

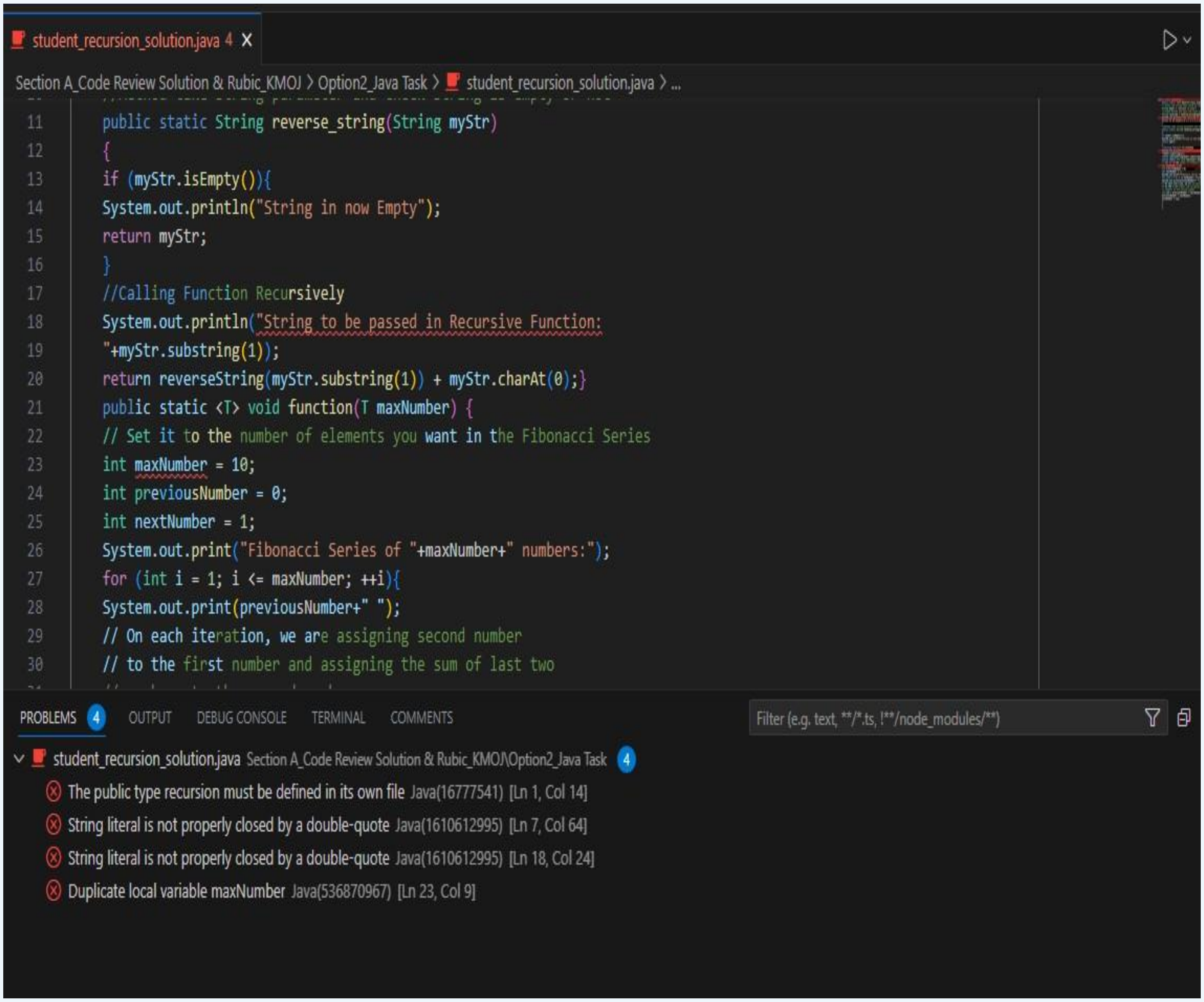
Student's recursion.java syntax

This code has errors because:

1. The Literal strings are not properly formatted and concatenated.

2. The one of the opened curly braces is closed, line 19.

3. The generic code raised a duplicate type error because maxNumber was implicitly defined as any, line 20.



student_recursion_solution.java 4 X

Section A_Code Review Solution & Rubic_KMOJ > Option2_Java Task > student_recursion_solution.java > ...

```
11  public static String reverse_string(String myStr)
12  {
13  if (myStr.isEmpty()){
14  System.out.println("String in now Empty");
15  return myStr;
16  }
17  //Calling Function Recursively
18  System.out.println("String to be passed in Recursive Function:
19  "+myStr.substring(1));
20  return reverseString(myStr.substring(1)) + myStr.charAt(0);}
21  public static <T> void function(T maxNumber) {
22  // Set it to the number of elements you want in the Fibonacci Series
23  int maxNumber = 10;
24  int previousNumber = 0;
25  int nextNumber = 1;
26  System.out.print("Fibonacci Series of "+maxNumber+" numbers:");
27  for (int i = 1; i <= maxNumber; ++i){
28  System.out.print(previousNumber+" ");
29  // On each iteration, we are assigning second number
30  // to the first number and assigning the sum of last two
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

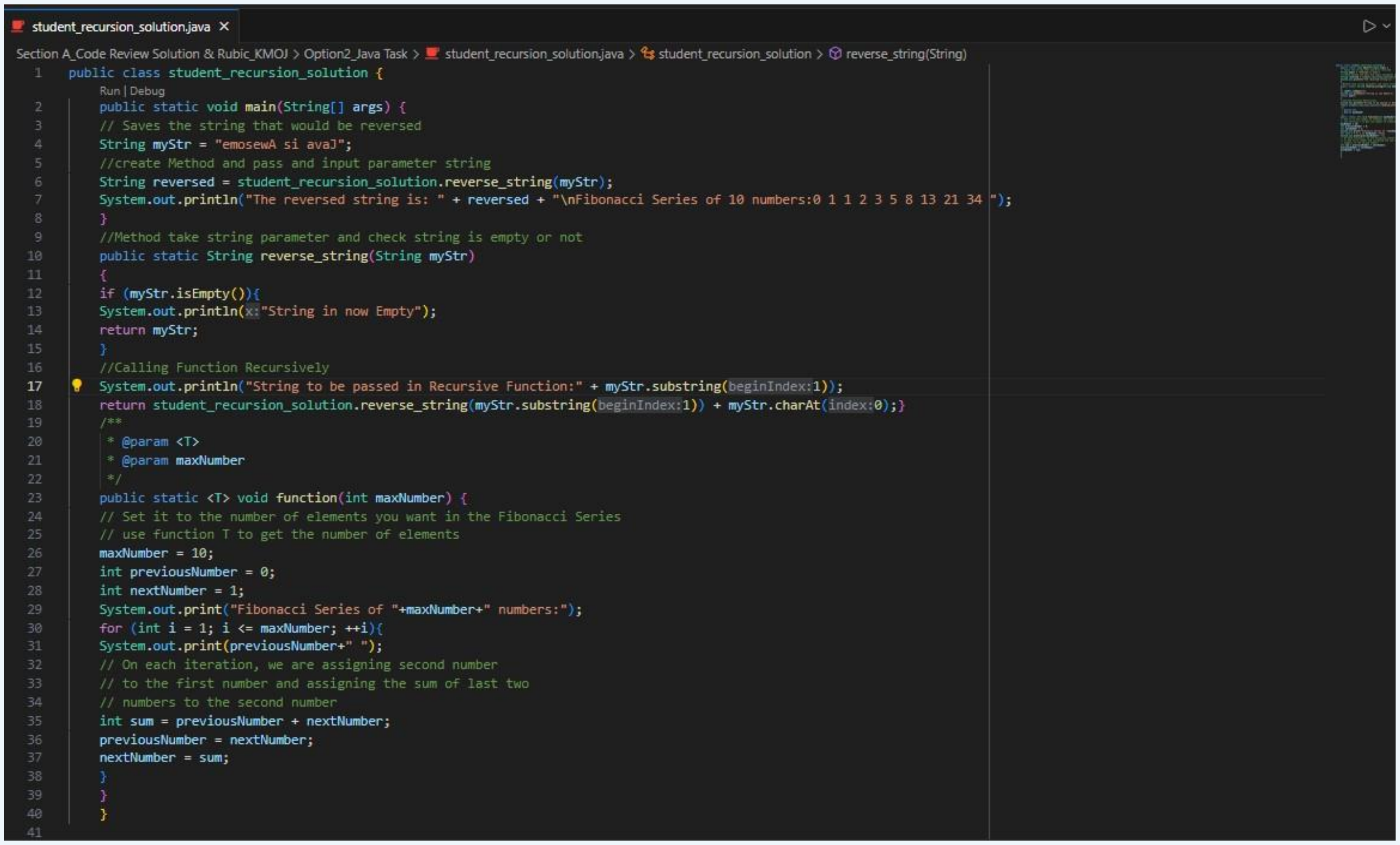
```
student_recursion_solution.java Section A_Code Review Solution & Rubic_KMOJ\Option2_Java Task 4
  The public type recursion must be defined in its own file Java(16777541) [Ln 1, Col 14]
  String literal is not properly closed by a double-quote Java(1610612995) [Ln 7, Col 64]
  String literal is not properly closed by a double-quote Java(1610612995) [Ln 18, Col 24]
  Duplicate local variable maxNumber Java(536870967) [Ln 23, Col 9]
```

Steps on how to fix the error in the student’s recursion.java file:

1. Clear the Literal String errors in the code.

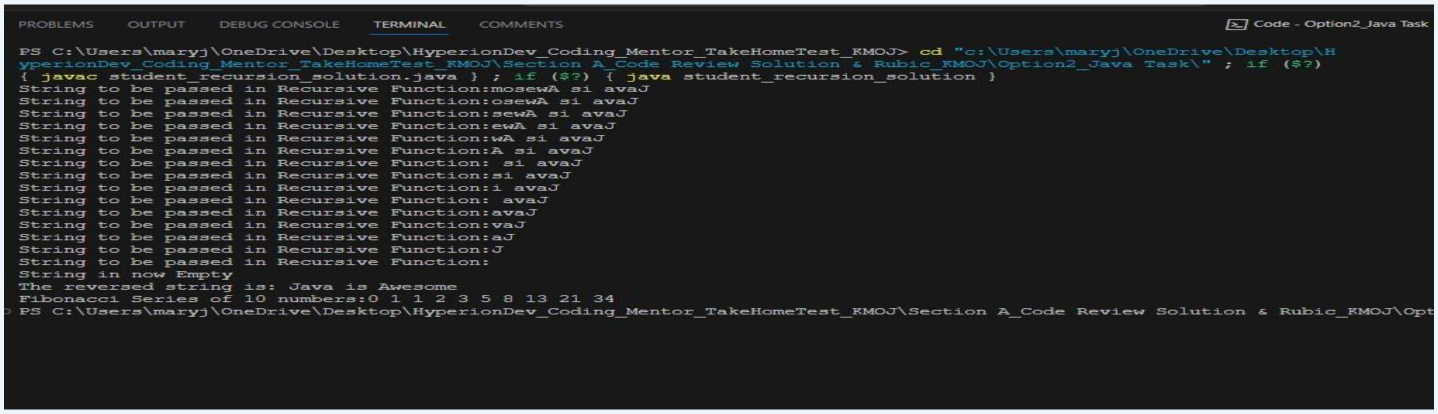
2. Then clear the type error raised on public static function<T>, let maxNumber be assigned as an Int, Line 23.

3. Modify the output display to be less cumbersome and make it more meaningful to appeal to the user and increase user experience.


student_recursion_solution.java
Section A_Code Review Solution & Rubic_KMOJ > Option2_Java Task > student_recursion_solution.java > student_recursion_solution > reverse_string(String)
Run | Debug

```java
1   public class student_recursion_solution {
2       public static void main(String[] args) {
3       // Saves the string that would be reversed
4       String myStr = "emosewA si avaJ";
5       //create Method and pass and input parameter string
6       String reversed = student_recursion_solution.reverse_string(myStr);
7       System.out.println("The reversed string is: " + reversed + "\nFibonacci Series of 10 numbers:0 1 1 2 3 5 8 13 21 34 ");
8       }
9       //Method take string parameter and check string is empty or not
10      public static String reverse_string(String myStr)
11      {
12      if (myStr.isEmpty()){
13      System.out.println(x:"String in now Empty");
14      return myStr;
15      }
16      //Calling Function Recursively
17      System.out.println("String to be passed in Recursive Function:" + myStr.substring(beginIndex:1));
18      return student_recursion_solution.reverse_string(myStr.substring(beginIndex:1)) + myStr.charAt(index:0);}
19      /**
20       * @param <T>
21       * @param maxNumber
22       */
23      public static <T> void function(int maxNumber) {
24      // Set it to the number of elements you want in the Fibonacci Series
25      // use function T to get the number of elements
26      maxNumber = 10;
27      int previousNumber = 0;
28      int nextNumber = 1;
29      System.out.print("Fibonacci Series of "+maxNumber+" numbers:");
30      for (int i = 1; i <= maxNumber; ++i){
31      System.out.print(previousNumber+" ");
32      // On each iteration, we are assigning second number
33      // to the first number and assigning the sum of last two
34      // numbers to the second number
35      int sum = previousNumber + nextNumber;
36      previousNumber = nextNumber;
37      nextNumber = sum;
38      }
39      }
40      }
41
```

Shown below is the student’s recursion.java program output after all the above stated errors were cleared and fixed.



```
PS C:\Users\maryj\OneDrive\Desktop\HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ> cd "c:\Users\maryj\OneDrive\Desktop\H
yperionDev_Coding_Mentor_TakeHomeTest_KMOJ\Section A_Code Review Solution & Rubic_KMOJ\Option2_Java Task\" ; if ($?)
{ javac student_recursion_solution.java } ; if ($?) { java student_recursion_solution }
String to be passed in Recursive Function:mosewA si avaJ
String to be passed in Recursive Function:osewA si avaJ
String to be passed in Recursive Function:sewA si avaJ
String to be passed in Recursive Function:ewA si avaJ
String to be passed in Recursive Function:wA si avaJ
String to be passed in Recursive Function:A si avaJ
String to be passed in Recursive Function: si avaJ
String to be passed in Recursive Function:si avaJ
String to be passed in Recursive Function:i avaJ
String to be passed in Recursive Function: avaJ
String to be passed in Recursive Function:avaJ
String to be passed in Recursive Function:vaJ
String to be passed in Recursive Function:aJ
String to be passed in Recursive Function:J
String to be passed in Recursive Function:
String in now Empty
The reversed string is: Java is Awesome
Fibonacci Series of 10 numbers:0 1 1 2 3 5 8 13 21 34
PS C:\Users\maryj\OneDrive\Desktop\HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ\Section A_Code Review Solution & Rubic_KMOJ\Opt
```

MENTOR’s EXAMPLE SOLUTION (INPUT)

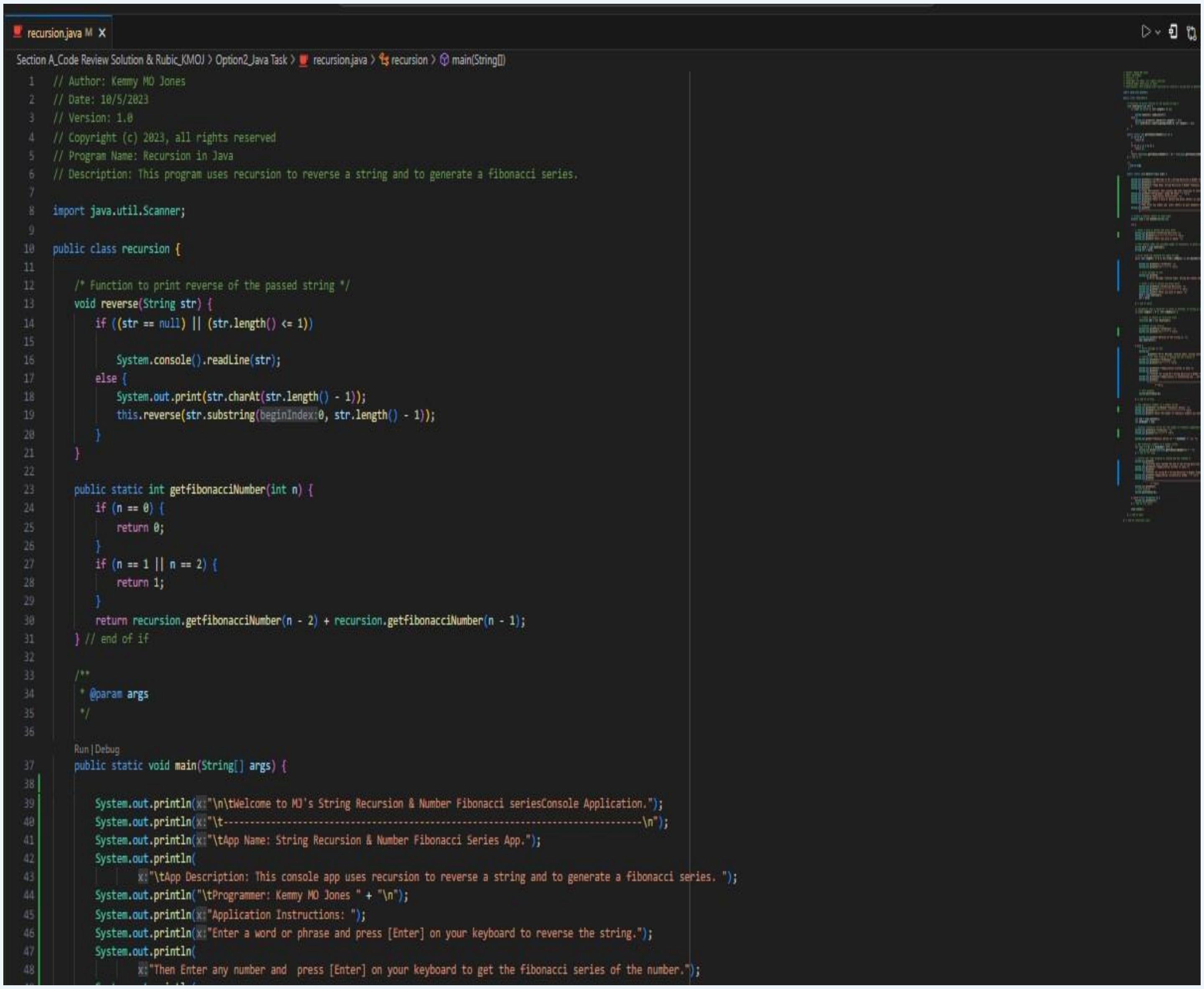
recursion.java file: improvement students can make on their program

1. **Prompt user for input and displays the anagram of the input string by using the scanner class.**

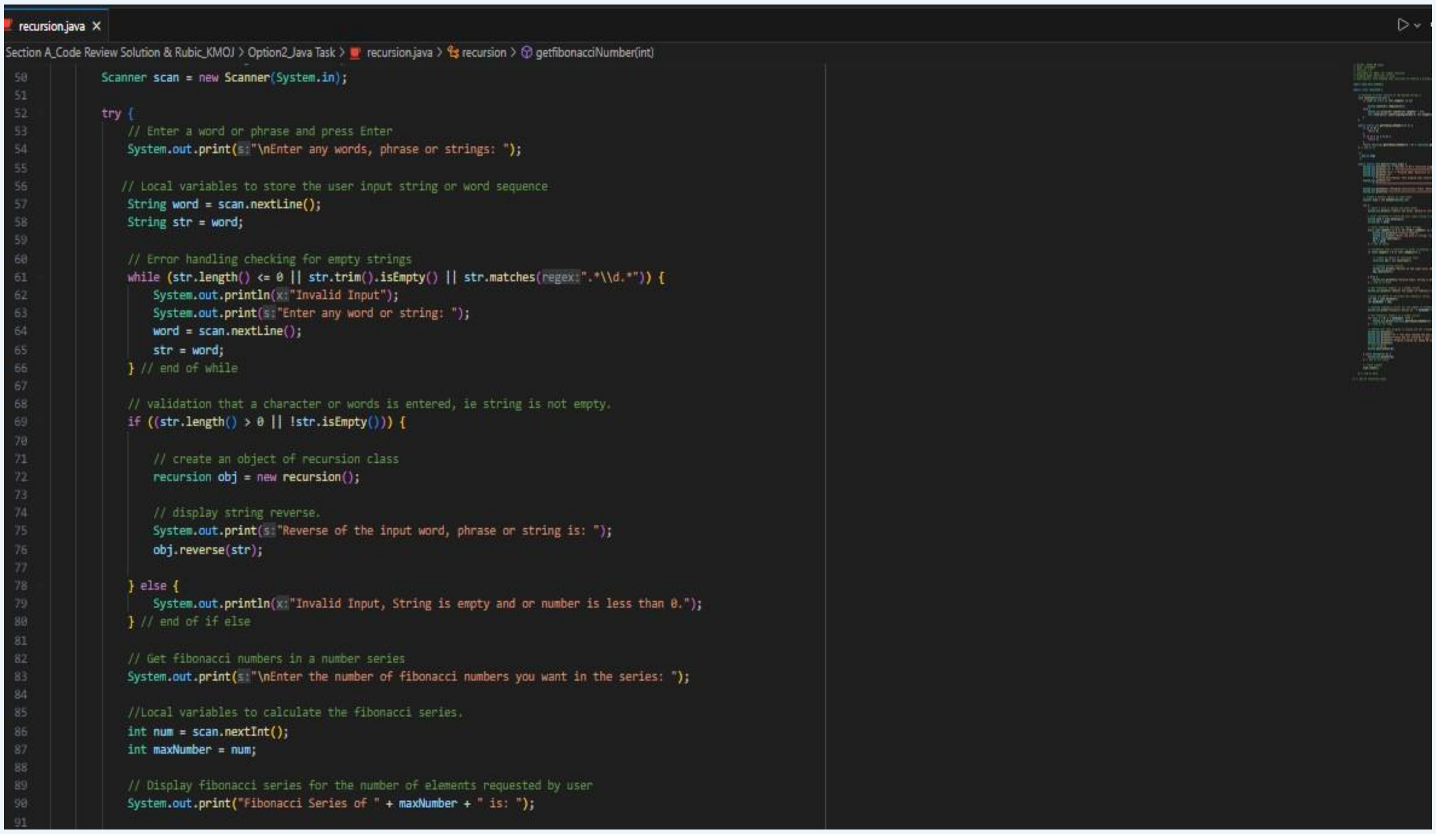
2. **Code is compartmentalized and modularized, has main method as the entry point, code validation and error handling for example when the input is an ‘empty string’, etc.**

3. **Add documentation and comments on important lines or chunk of code describing what the code does.**

4. **Add instruction and feedback to user on the task success or completion and error message display in order to create a better user experience.**


recursion.java M ×
Section A_Code Review Solution & Rubic_KMOJ › Option2_Java Task › recursion.java › recursion › main(String[])
Run | Debug

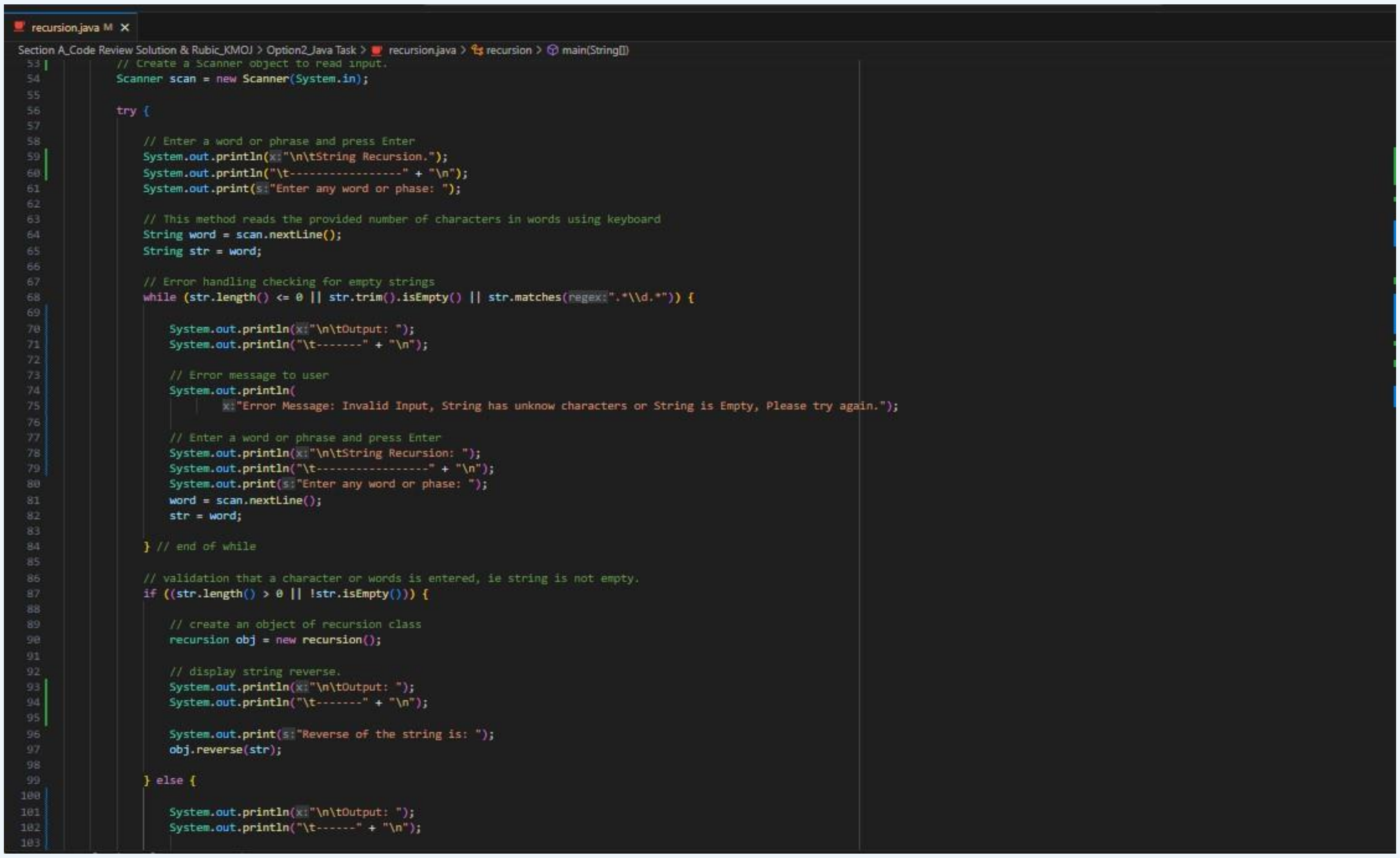
```java
// Author: Kemmy MO Jones
// Date: 10/5/2023
// Version: 1.0
// Copyright (c) 2023, all rights reserved
// Program Name: Recursion in Java
// Description: This program uses recursion to reverse a string and to generate a fibonacci series.

import java.util.Scanner;

public class recursion {

    /* Function to print reverse of the passed string */
    void reverse(String str) {
        if ((str == null) || (str.length() <= 1))

            System.console().readLine(str);
        else {
            System.out.print(str.charAt(str.length() - 1));
            this.reverse(str.substring(beginIndex:0, str.length() - 1));
        }
    }

    public static int getfibonacciNumber(int n) {
        if (n == 0) {
            return 0;
        }
        if (n == 1 || n == 2) {
            return 1;
        }
        return recursion.getfibonacciNumber(n - 2) + recursion.getfibonacciNumber(n - 1);
    } // end of if

    /**
     * @param args
     */

    public static void main(String[] args) {

        System.out.println(x:"\n\tWelcome to MJ's String Recursion & Number Fibonacci seriesConsole Application.");
        System.out.println(x:"\t---------------------------------------------------------------------------------\n");
        System.out.println(x:"\tApp Name: String Recursion & Number Fibonacci Series App.");
        System.out.println(
                x:"\tApp Description: This console app uses recursion to reverse a string and to generate a fibonacci series. ");
        System.out.println("\tProgrammer: Kemmy MO Jones " + "\n");
        System.out.println(x:"Application Instructions: ");
        System.out.println(x:"Enter a word or phrase and press [Enter] on your keyboard to reverse the string.");
        System.out.println(
                x:"Then Enter any number and  press [Enter] on your keyboard to get the fibonacci series of the number.");
```



recursion.java

Section A_Code Review Solution & Rubic_KMOJ > Option2_Java Task > recursion.java > recursion > getfibonacciNumber(int)

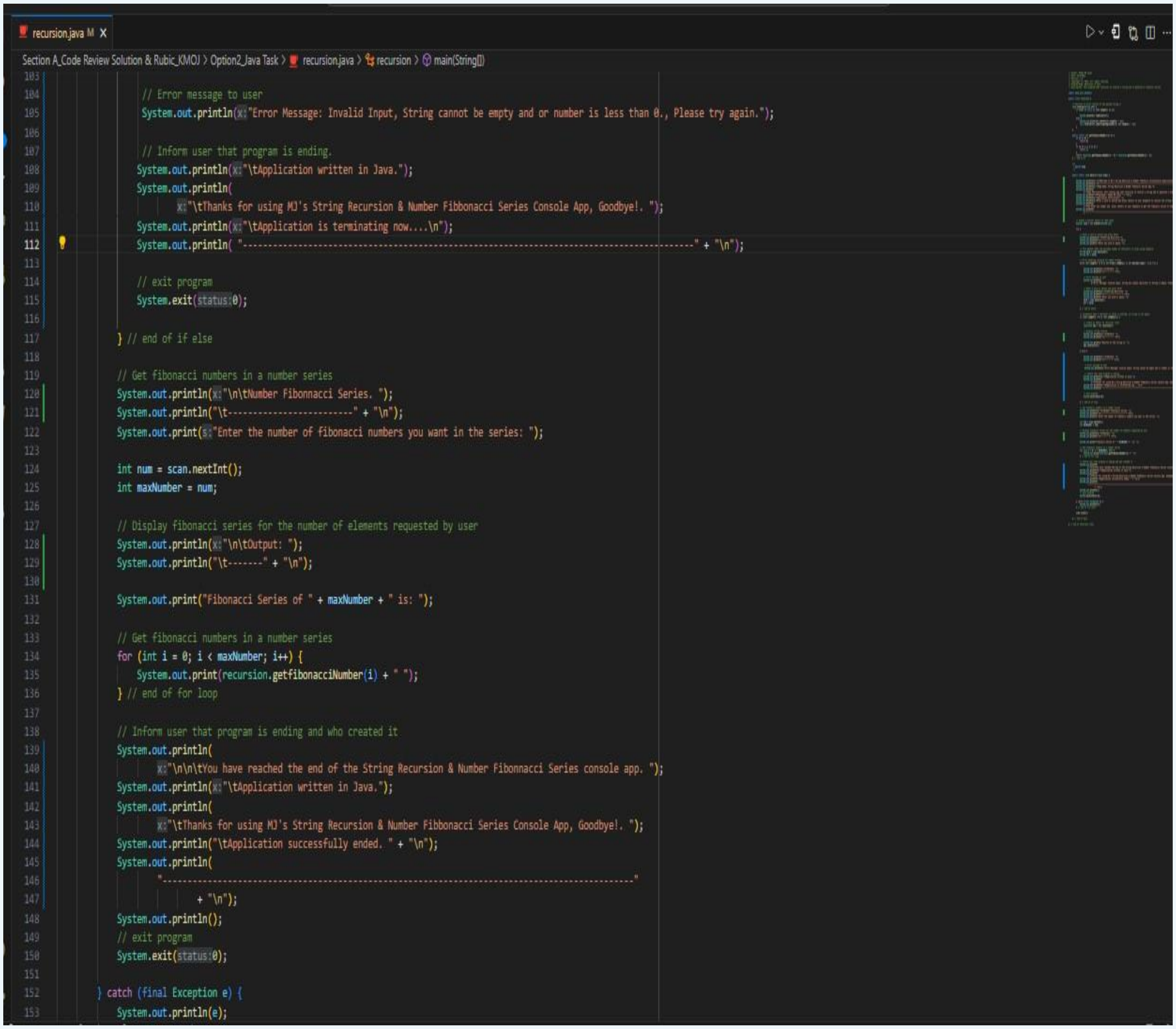
```java
        Scanner scan = new Scanner(System.in);

        try {
            // Enter a word or phrase and press Enter
            System.out.print(s:"\nEnter any words, phrase or strings: ");

          // Local variables to store the user input string or word sequence
            String word = scan.nextLine();
            String str = word;

            // Error handling checking for empty strings
            while (str.length() <= 0 || str.trim().isEmpty() || str.matches(regex:".*\\d.*")) {
                System.out.println(x:"Invalid Input");
                System.out.print(s:"Enter any word or string: ");
                word = scan.nextLine();
                str = word;
            } // end of while

            // validation that a character or words is entered, ie string is not empty.
            if ((str.length() > 0 || !str.isEmpty())) {

                // create an object of recursion class
                recursion obj = new recursion();

                // display string reverse.
                System.out.print(s:"Reverse of the input word, phrase or string is: ");
                obj.reverse(str);

            } else {
                System.out.println(x:"Invalid Input, String is empty and or number is less than 0.");
            } // end of if else

            // Get fibonacci numbers in a number series
            System.out.print(s:"\nEnter the number of fibonacci numbers you want in the series: ");

            //Local variables to calculate the fibonacci series.
            int num = scan.nextInt();
            int maxNumber = num;

            // Display fibonacci series for the number of elements requested by user
            System.out.print("Fibonacci Series of " + maxNumber + " is: ");

```



recursion.java M

Section A_Code Review Solution & Rubic_KMOJ > Option2_Java Task > recursion.java > recursion > main(String[])

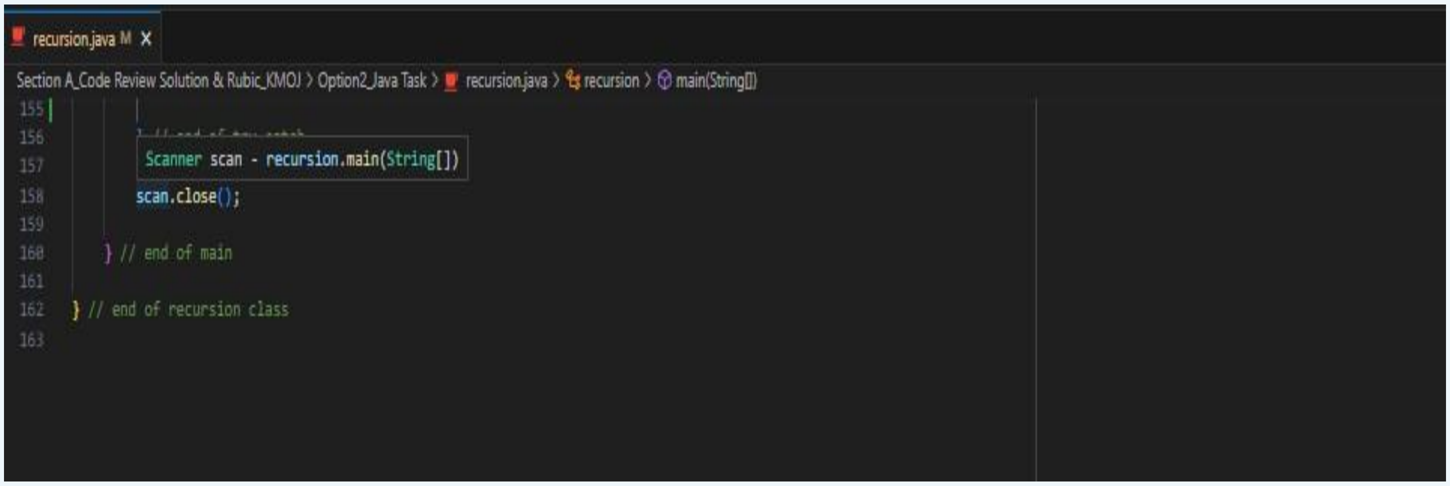
```java
        // Create a Scanner object to read input.
        Scanner scan = new Scanner(System.in);

        try {

            // Enter a word or phrase and press Enter
            System.out.println(x:"\n\tString Recursion.");
            System.out.println("\t----------------" + "\n");
            System.out.print(s:"Enter any word or phase: ");

            // This method reads the provided number of characters in words using keyboard
            String word = scan.nextLine();
            String str = word;

            // Error handling checking for empty strings
            while (str.length() <= 0 || str.trim().isEmpty() || str.matches(regex:".*\\d.*")) {

                System.out.println(x:"\n\tOutput: ");
                System.out.println("\t-------" + "\n");

                // Error message to user
                System.out.println(
                        x:"Error Message: Invalid Input, String has unknow characters or String is Empty, Please try again.");

                // Enter a word or phrase and press Enter
                System.out.println(x:"\n\tString Recursion: ");
                System.out.println("\t-----------------" + "\n");
                System.out.print(s:"Enter any word or phase: ");
                word = scan.nextLine();
                str = word;

            } // end of while

            // validation that a character or words is entered, ie string is not empty.
            if ((str.length() > 0 || !str.isEmpty())) {

                // create an object of recursion class
                recursion obj = new recursion();

                // display string reverse.
                System.out.println(x:"\n\tOutput: ");
                System.out.println("\t-------" + "\n");

                System.out.print(s:"Reverse of the string is: ");
                obj.reverse(str);

            } else {

                System.out.println(x:"\n\tOutput: ");
                System.out.println("\t------" + "\n");

```



recursion.java M

Section A_Code Review Solution & Rubic_KMOJ > Option2_Java Task > recursion.java > recursion > main(String[])

```java
                    // Error message to user
                    System.out.println(x:"Error Message: Invalid Input, String cannot be empty and or number is less than 0., Please try again.");

                    // Inform user that program is ending.
                   System.out.println(x:"\tApplication written in Java.");
                   System.out.println(
                           x:"\tThanks for using MJ's String Recursion & Number Fibbonacci Series Console App, Goodbye!. ");
                   System.out.println(x:"\tApplication is terminating now....\n");
                   System.out.println( "----------------------------------------------------------------------------------" + "\n");

                   // exit program
                   System.exit(status:0);

               } // end of if else

               // Get fibonacci numbers in a number series
               System.out.println(x:"\n\tNumber Fibonnacci Series. ");
               System.out.println("\t-----------------------" + "\n");
               System.out.print(s:"Enter the number of fibonacci numbers you want in the series: ");

               int num = scan.nextInt();
               int maxNumber = num;

               // Display fibonacci series for the number of elements requested by user
               System.out.println(x:"\n\tOutput: ");
               System.out.println("\t-------" + "\n");

               System.out.print("Fibonacci Series of " + maxNumber + " is: ");

               // Get fibonacci numbers in a number series
               for (int i = 0; i < maxNumber; i++) {
                   System.out.print(recursion.getfibonacciNumber(i) + " ");
               } // end of for loop

               // Inform user that program is ending and who created it
               System.out.println(
                       x:"\n\n\tYou have reached the end of the String Recursion & Number Fibonnacci Series console app. ");
               System.out.println(x:"\tApplication written in Java.");
               System.out.println(
                       x:"\tThanks for using MJ's String Recursion & Number Fibbonacci Series Console App, Goodbye!. ");
               System.out.println("\tApplication successfully ended. " + "\n");
               System.out.println(
                       "----------------------------------------------------------------------------------"
                               + "\n");
               System.out.println();
               // exit program
               System.exit(status:0);

           } catch (final Exception e) {
               System.out.println(e);
```



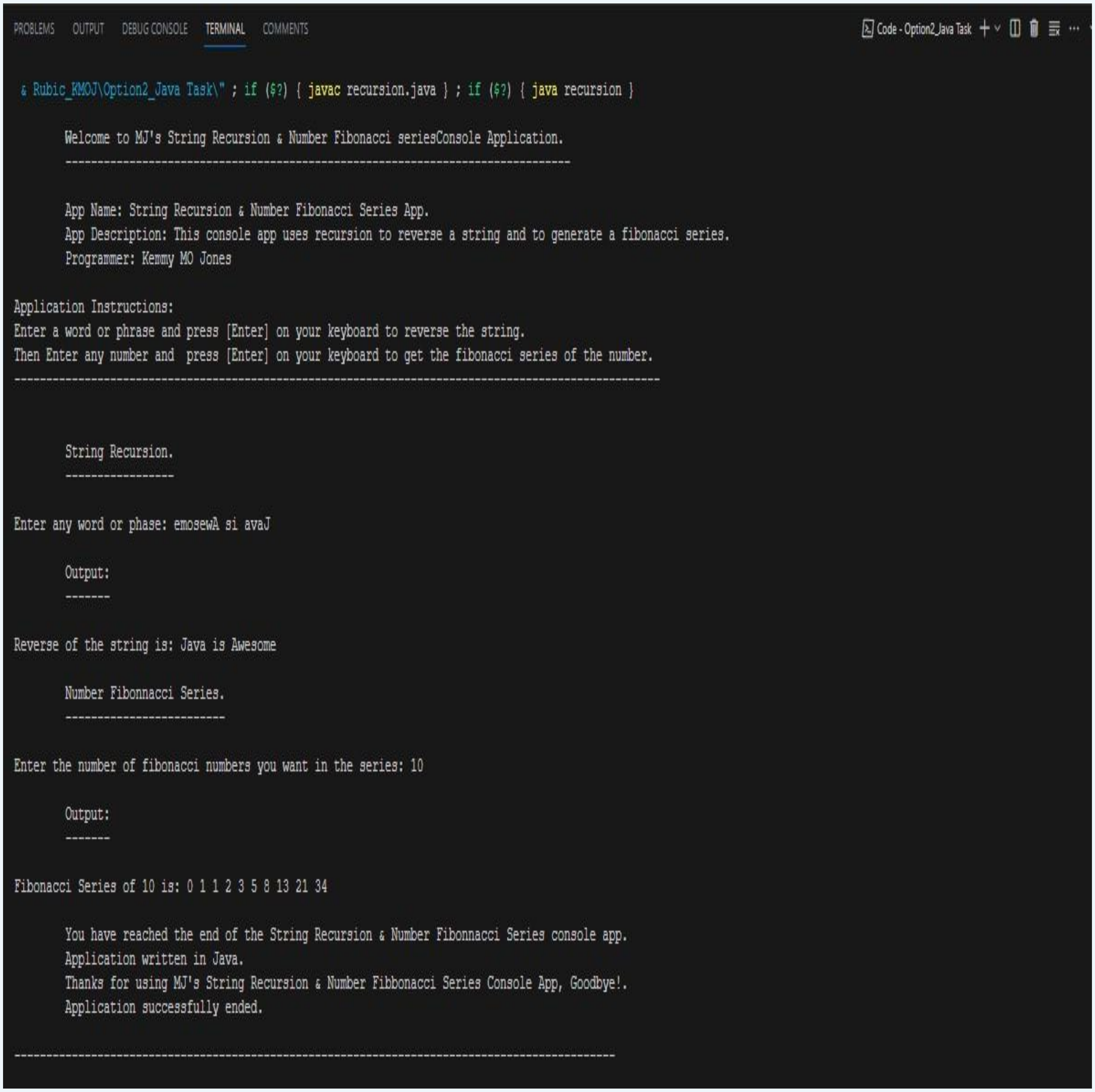
recursion.java M

Section A_Code Review Solution & Rubic_KMOJ > Option2_Java Task > recursion.java > recursion > main(String[])

```java
            } // end of try catch
            Scanner scan - recursion.main(String[])
            scan.close();

        } // end of main

    } // end of recursion class
```

MENTOR's EXAMPLE SOLUTION (OUTPUT)

recursion.java file: The recursion console app output is displayed as shown below:



```
 & Rubic_KMOJ\Option2_Java Task\" ; if ($?) { javac recursion.java } ; if ($?) { java recursion }

        Welcome to MJ's String Recursion & Number Fibonacci seriesConsole Application.
        ------------------------------------------------------------------------------

        App Name: String Recursion & Number Fibonacci Series App.
        App Description: This console app uses recursion to reverse a string and to generate a fibonacci series.
        Programmer: Kemmy MO Jones

Application Instructions:
Enter a word or phrase and press [Enter] on your keyboard to reverse the string.
Then Enter any number and  press [Enter] on your keyboard to get the fibonacci series of the number.
----------------------------------------------------------------------------------------------------



        String Recursion.
        -----------------

Enter any word or phase: emosewA si avaJ

        Output:
        -------

Reverse of the string is: Java is Awesome

        Number Fibonacci Series.
        ------------------------

Enter the number of fibonacci numbers you want in the series: 10

        Output:
        -------

Fibonacci Series of 10 is: 0 1 1 2 3 5 8 13 21 34

        You have reached the end of the String Recursion & Number Fibonnacci Series console app.
        Application written in Java.
        Thanks for using MJ's String Recursion & Number Fibbonacci Series Console App, Goodbye!.
        Application successfully ended.

---------------------------------------------------------------------------------------------
```

Option 3: Ruby Task (number_palindrome.rb or number_palindrome.arb)						
CODE REVIEW RUBIC	GRADE & MARKS 10 Marks Each			CODE LINE	ERRORS / ISSUES TO RESOLVE	MENTOR'S REMARK
Code Correctness:	~Correct file name: ~ Displays Output: ~ Clean Coding: Total Marks (30): Grade:	Yes No Yes => =>	10 0 10 20 C+	4 -28	Errors due to: ~EOF error raised one of def function was not closed with 'end'. ~Error handling is not correct, gives a run time error, the output is always false	~Fix and clear all error raised because of unassigned or wrongfully assigned variables. ~Ensure all opened functions are closed with end to avoid memory leak and EOF (end of file) reached error. ~ Fix the if's statements and the while loops are closed as it is not checking the palindrome if it is true or false correctly.
Efficiency:	~Code Reliability: ~ Speed to compile: ~ Methodology: Total Marks (30): Grade:	Fair Excl. Good => =>	4 10 3 17 C	4 - 28	~Code is prone to error, due to implicit variable definition. ~The code does not adequately pass for a pseudocode algorithm ~Runtime error because of incorrect error handling.	~It is good practice to always have a main () method which is an entry point and display the output of your program resulting in a neater, and the input codes in modules such as functions, methods this helps enhance the code and result in a neater organized reusable code. ~It is good practice to let your program accept user input, which is then processed, and the compiler displays the desired output it enhances user experience. ~It is good practice to use functions and methods to compartmentalize code thus modularizing codes enhance readability of code syntax and aids easy debugging of code in case of any error during code compile time or run time.
Style:	~Indentation: ~Alignment: ~Name Convention: Total Marks (30): Grade:	Yes Yes Excl => =>	10 10 10 30 A	1-29	To Learn more, read this: Ruby Conversion Methods.[vi] Ruby Conversion Methods.[vii]	~It is good practice to let Class name match the file name. It helps you or any programmer assigned to maintain the codes to be able to identify which class is called in case your code contains more than one class. ~Be extra careful when using implicitly defining variables may cause unwanted code leak, compile error or runtime errors
Documentation	~Comments: ~Documentation: Total Marks (10): Grade:	Yes Yes => =>	5 3 8 A	1, 6-10, 16-17, 21-22	~Documentation can be improved.	~It is good practice to include comments and documentation in a program it enhances the code to be more comprehensive, easy to understand what each code does and makes code debugging, improvement easier.
Possible Improvement to Enhance Code	-Fix and clear EOF error by closing the if-else statement on line 4-13. -It is good practice to break your codes into simple functional set of code chunks or modularized with methods and functions, makes it neater, readable and helps make debugging easy. -Ensure to call necessary parameters, explicit definition of variables and careful use of generic coding when necessary but do not use always. -It is very good practice to create a program that takes input from users, easy to use by users and leaving user instructions on how to proceed when using the program. -It is good practice to add error handling and input validation by correcting the if-else statement. References[viii]					

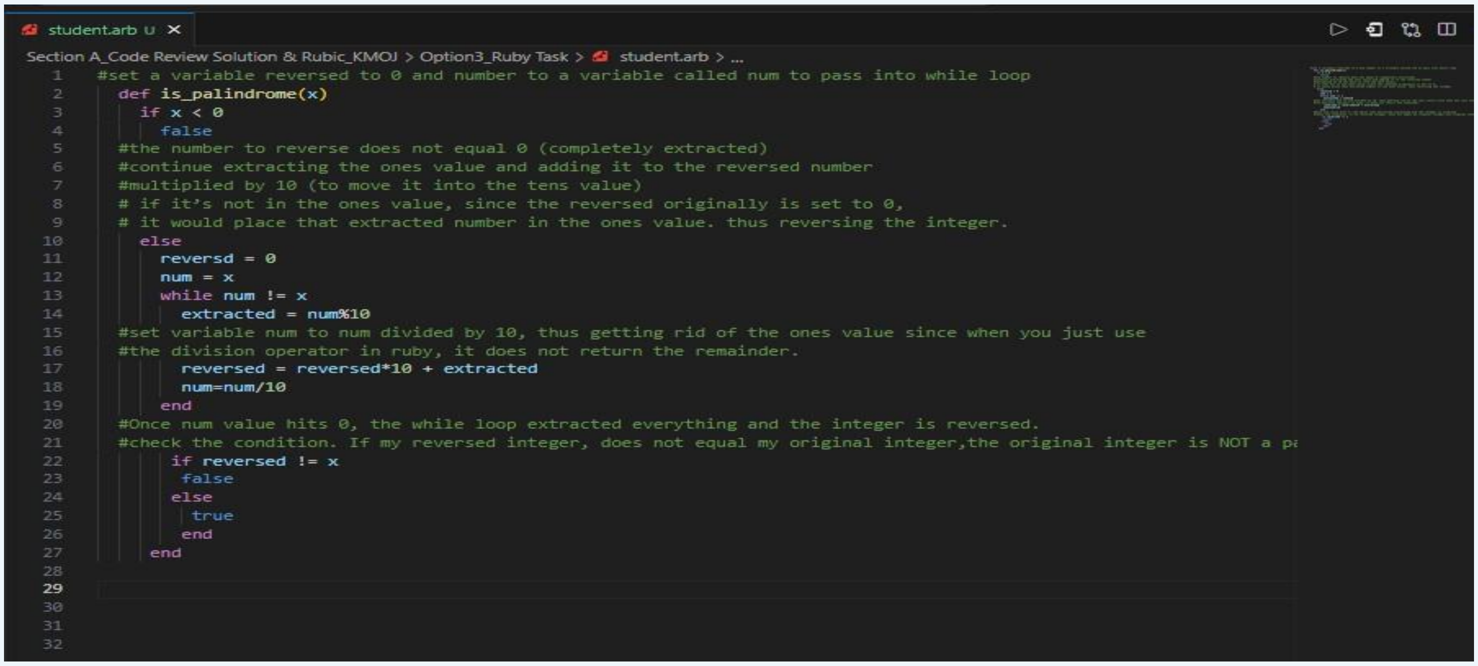
Student's number_palindrome.rb Ruby Algorithm

This code has errors because:

1. Displays EOF error because one of the ‘if-else’ statement is not closed by an end -tag, line 4-13

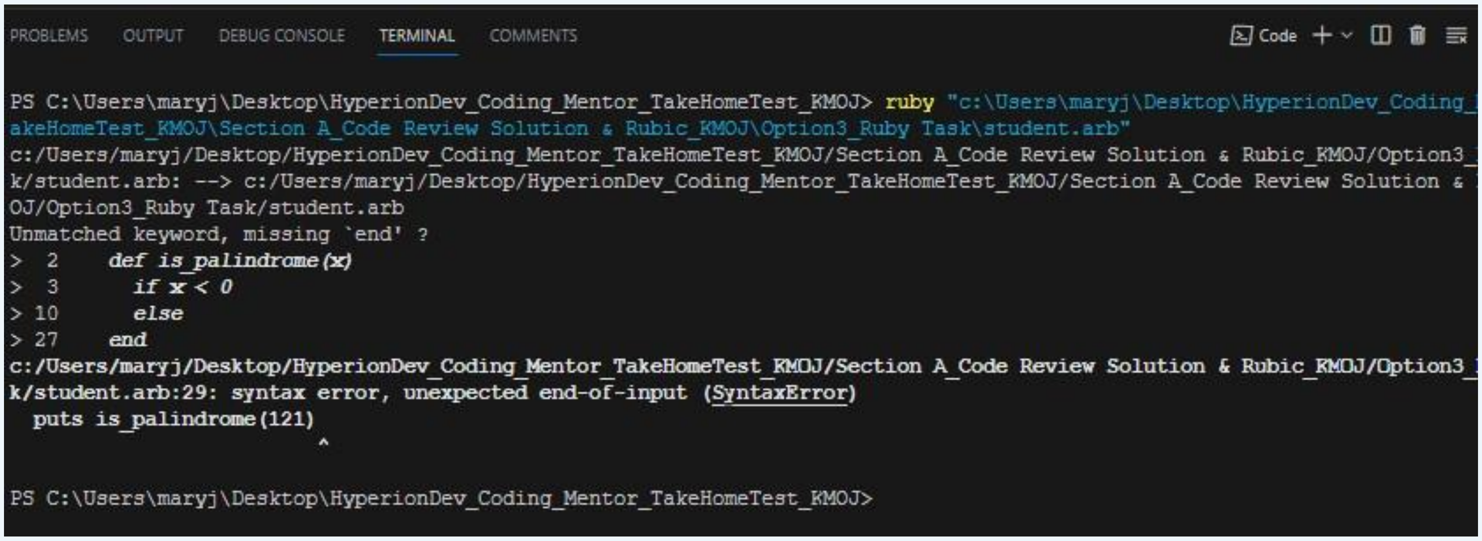
2. The Ruby algorithm pseudocode is adequately correct thus causing the code to incomplete.

3. The program has no output to test it and the error handling and validation's syntax is not correct hence causing Runtime error where all number is displayed as False .



student.arb U ×
Section A_Code Review Solution & Rubic_KMOJ > Option3_Ruby Task > student.arb > ...

```
 1  #set a variable reversed to 0 and number to a variable called num to pass into while loop
 2    def is_palindrome(x)
 3      if x < 0
 4        false
 5    #the number to reverse does not equal 0 (completely extracted)
 6    #continue extracting the ones value and adding it to the reversed number
 7    #multiplied by 10 (to move it into the tens value)
 8    # if it’s not in the ones value, since the reversed originally is set to 0,
 9    # it would place that extracted number in the ones value. thus reversing the integer.
10      else
11        reversd = 0
12        num = x
13        while num != x
14          extracted = num%10
15    #set variable num to num divided by 10, thus getting rid of the ones value since when you just use
16    #the division operator in ruby, it does not return the remainder.
17          reversed = reversed*10 + extracted
18          num=num/10
19        end
20    #Once num value hits 0, the while loop extracted everything and the integer is reversed.
21    #check the condition. If my reversed integer, does not equal my original integer,the original integer is NOT a pa
22           if reversed != x
23            false
24           else
25             true
26            end
27       end
28
29
30
31
32
```



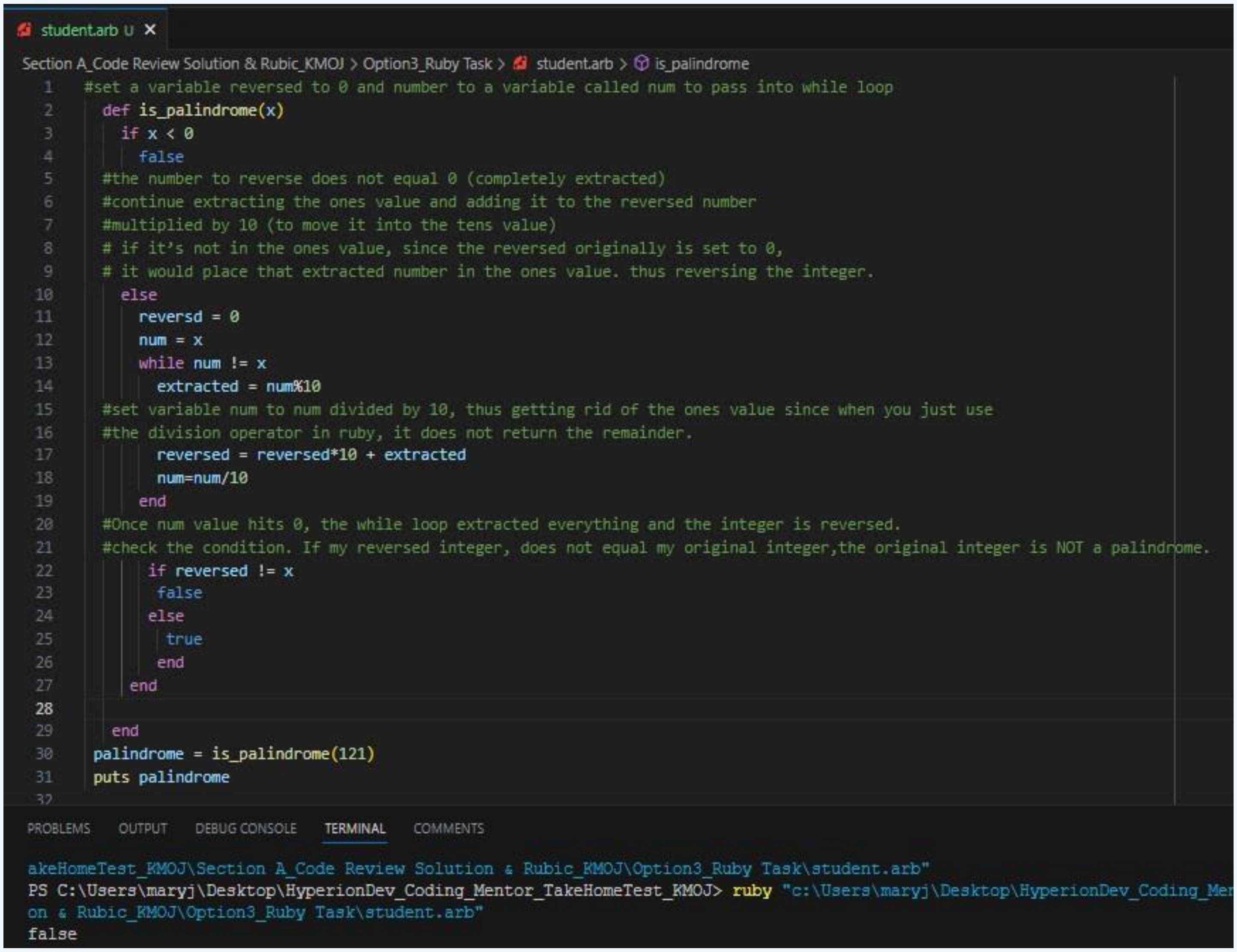
```
PS C:\Users\maryj\Desktop\HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ> ruby "c:\Users\maryj\Desktop\HyperionDev_Coding_
akeHomeTest_KMOJ\Section A_Code Review Solution & Rubic_KMOJ\Option3_Ruby Task\student.arb"
c:/Users/maryj/Desktop/HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ/Section A_Code Review Solution & Rubic_KMOJ/Option3_
k/student.arb: --> c:/Users/maryj/Desktop/HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ/Section A_Code Review Solution &
OJ/Option3_Ruby Task/student.arb
Unmatched keyword, missing `end' ?
>  2    def is_palindrome(x)
>  3      if x < 0
> 10      else
> 27    end
c:/Users/maryj/Desktop/HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ/Section A_Code Review Solution & Rubic_KMOJ/Option3_
k/student.arb:29: syntax error, unexpected end-of-input (SyntaxError)
  puts is_palindrome(121)
                        ^

PS C:\Users\maryj\Desktop\HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ>
```

Steps on how to fix the error in the student’s number_palidrome.rb file:

1. Close all the open function with an 'end' tag.

2. Enhance the if-else statement to checked if the extracted number is a palindrome or not and check if the input is positive number or not.

3. Add output to test the function and modify the output display to be less cumbersome and make it more meaningful to appeal to the user and increase user experience.


student.arb U ✕
Section A_Code Review Solution & Rubic_KMOJ > Option3_Ruby Task > student.arb > is_palindrome
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
COMMENTS

```ruby
#set a variable reversed to 0 and number to a variable called num to pass into while loop
  def is_palindrome(x)
    if x < 0
      false
  #the number to reverse does not equal 0 (completely extracted)
  #continue extracting the ones value and adding it to the reversed number
  #multiplied by 10 (to move it into the tens value)
  # if it’s not in the ones value, since the reversed originally is set to 0,
  # it would place that extracted number in the ones value. thus reversing the integer.
    else
      reversd = 0
      num = x
      while num != x
        extracted = num%10
  #set variable num to num divided by 10, thus getting rid of the ones value since when you just use
  #the division operator in ruby, it does not return the remainder.
        reversed = reversed*10 + extracted
        num=num/10
      end
  #Once num value hits 0, the while loop extracted everything and the integer is reversed.
  #check the condition. If my reversed integer, does not equal my original integer,the original integer is NOT a palindrome.
       if reversed != x
        false
       else
         true
        end
     end

   end
 palindrome = is_palindrome(121)
 puts palindrome
```

```
akeHomeTest_KMOJ\Section A_Code Review Solution & Rubic_KMOJ\Option3_Ruby Task\student.arb"
PS C:\Users\maryj\Desktop\HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ> ruby "c:\Users\maryj\Desktop\HyperionDev_Coding_Men
on & Rubic_KMOJ\Option3_Ruby Task\student.arb"
false
```

The above shows the student’s number Palindrome program output after all the above stated errors were cleared and fixed.

MENTOR's EXAMPLE SOLUTION (INPUT)

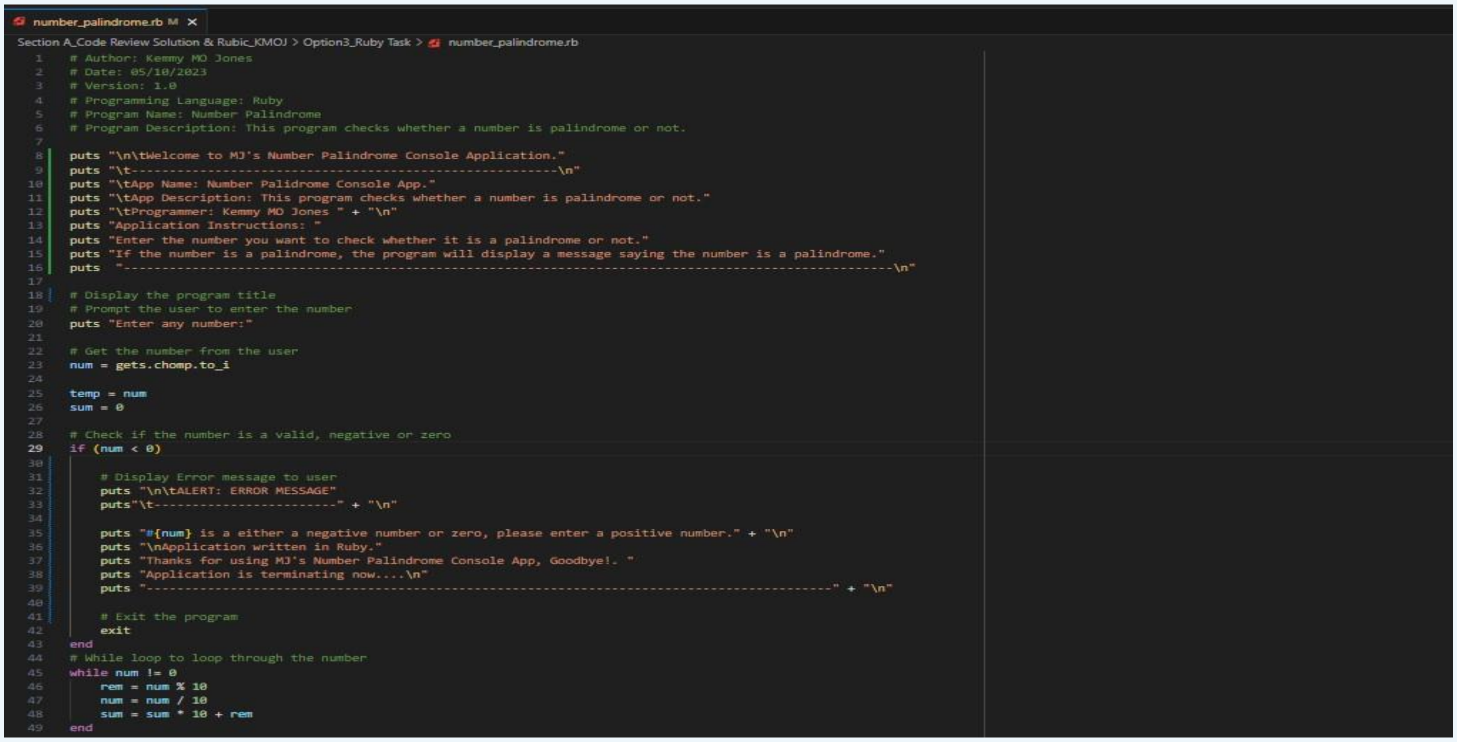
number_palindrome.rb file: improvements students can make on their program

1. Prompt user for input and displays the anagram of the input string by using the scanner class.

2. Code is compartmentalized and modularized, has main method as the entry point, code validation and error handling for example imputing empty string, etc.

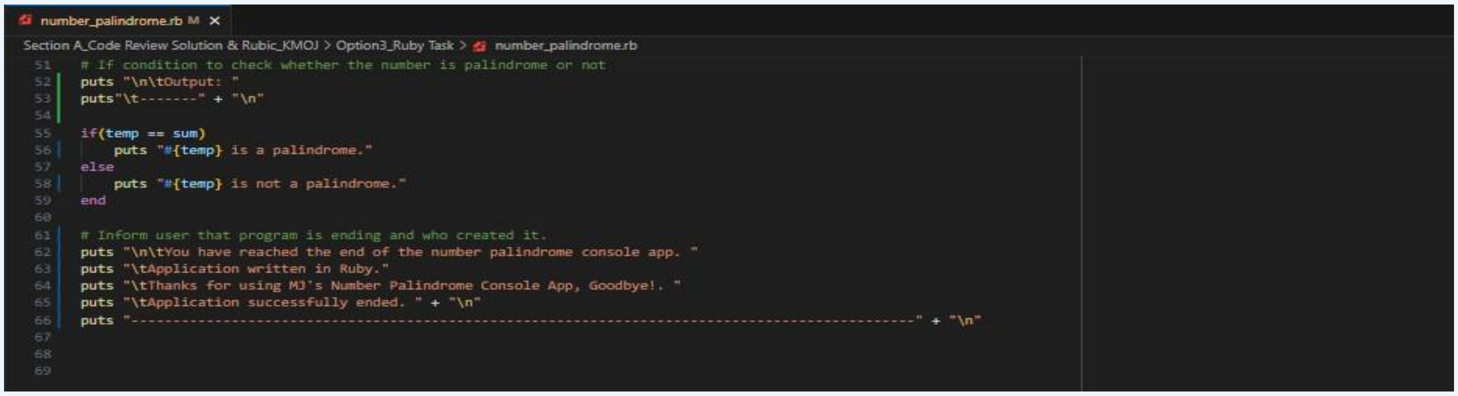
3. Add documentation and comments on important lines or chunk of code describing what the code does.

4. Add instruction and feedback to user on the task success or completion and error message display in order to create a better user experience.



number_palindrome.rb M ✕
Section A_Code Review Solution & Rubic_KMOJ > Option3_Ruby Task > number_palindrome.rb

```ruby
# Author: Kemmy MO Jones
# Date: 05/10/2023
# Version: 1.0
# Programming Language: Ruby
# Program Name: Number Palindrome
# Program Description: This program checks whether a number is palindrome or not.

puts "\n\tWelcome to MJ's Number Palindrome Console Application."
puts "\t-------------------------------------------------------\n"
puts "\tApp Name: Number Palidrome Console App."
puts "\tApp Description: This program checks whether a number is palindrome or not."
puts "\tProgrammer: Kemmy MO Jones " + "\n"
puts "Application Instructions: "
puts "Enter the number you want to check whether it is a palindrome or not."
puts "If the number is a palindrome, the program will display a message saying the number is a palindrome."
puts  "-----------------------------------------------------------------------------------------------------\n"

# Display the program title
# Prompt the user to enter the number
puts "Enter any number:"

# Get the number from the user
num = gets.chomp.to_i

temp = num
sum = 0

# Check if the number is a valid, negative or zero
if (num < 0)

    # Display Error message to user
    puts "\n\tALERT: ERROR MESSAGE"
    puts"\t-----------------------" + "\n"

    puts "#{num} is a either a negative number or zero, please enter a positive number." + "\n"
    puts "\nApplication written in Ruby."
    puts "Thanks for using MJ's Number Palindrome Console App, Goodbye!. "
    puts "Application is terminating now....\n"
    puts "-----------------------------------------------------------------------------------------" + "\n"

    # Exit the program
    exit
end
# While loop to loop through the number
while num != 0
    rem = num % 10
    num = num / 10
    sum = sum * 10 + rem
end
```

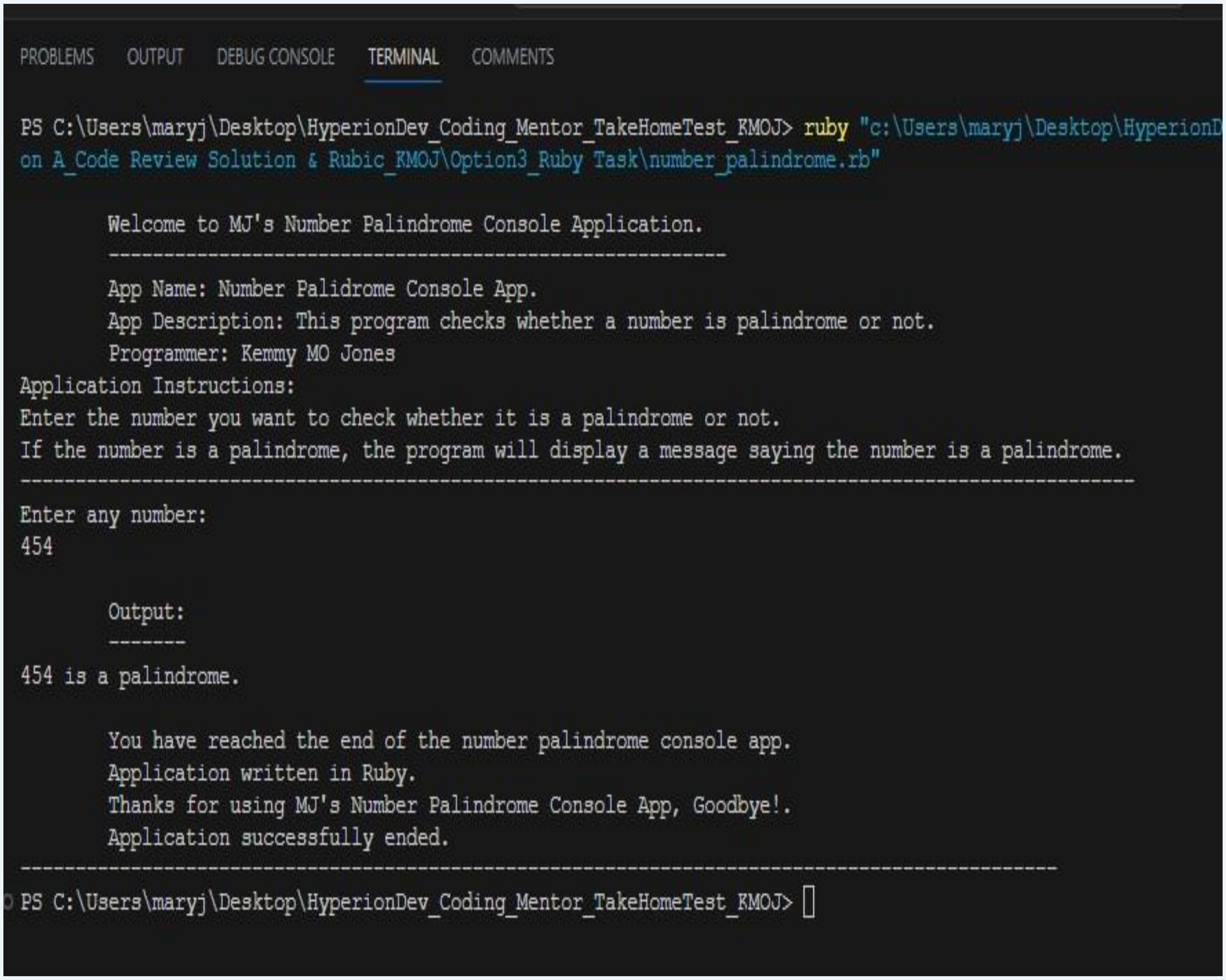


number_palindrome.rb M ✕
Section A_Code Review Solution & Rubic_KMOJ > Option3_Ruby Task > number_palindrome.rb

```ruby
# If condition to check whether the number is palindrome or not
puts "\n\tOutput: "
puts"\t-------" + "\n"

if(temp == sum)
    puts "#{temp} is a palindrome."
else
    puts "#{temp} is not a palindrome."
end

# Inform user that program is ending and who created it.
puts "\n\tYou have reached the end of the number palindrome console app. "
puts "\tApplication written in Ruby."
puts "\tThanks for using MJ's Number Palindrome Console App, Goodbye!. "
puts "\tApplication successfully ended. " + "\n"
puts "-----------------------------------------------------------------------------------------------" + "\n"
```

MENTOR's EXAMPLE SOLUTION (OUTPUT)

number_palindrome.rb file: The palindrome console app output is shown below



```
PS C:\Users\maryj\Desktop\HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ> ruby "c:\Users\maryj\Desktop\HyperionD
on A_Code Review Solution & Rubic_KMOJ\Option3_Ruby Task\number_palindrome.rb"

        Welcome to MJ's Number Palindrome Console Application.
        --------------------------------------------------------
        App Name: Number Palidrome Console App.
        App Description: This program checks whether a number is palindrome or not.
        Programmer: Kemmy MO Jones
Application Instructions:
Enter the number you want to check whether it is a palindrome or not.
If the number is a palindrome, the program will display a message saying the number is a palindrome.
----------------------------------------------------------------------------------------------------
Enter any number:
454

        Output:
        -------
454 is a palindrome.

        You have reached the end of the number palindrome console app.
        Application written in Ruby.
        Thanks for using MJ's Number Palindrome Console App, Goodbye!.
        Application successfully ended.
---------------------------------------------------------------------------------------------------
PS C:\Users\maryj\Desktop\HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ> 
```

Option 4: TypeScript Task (ceasar.ts)

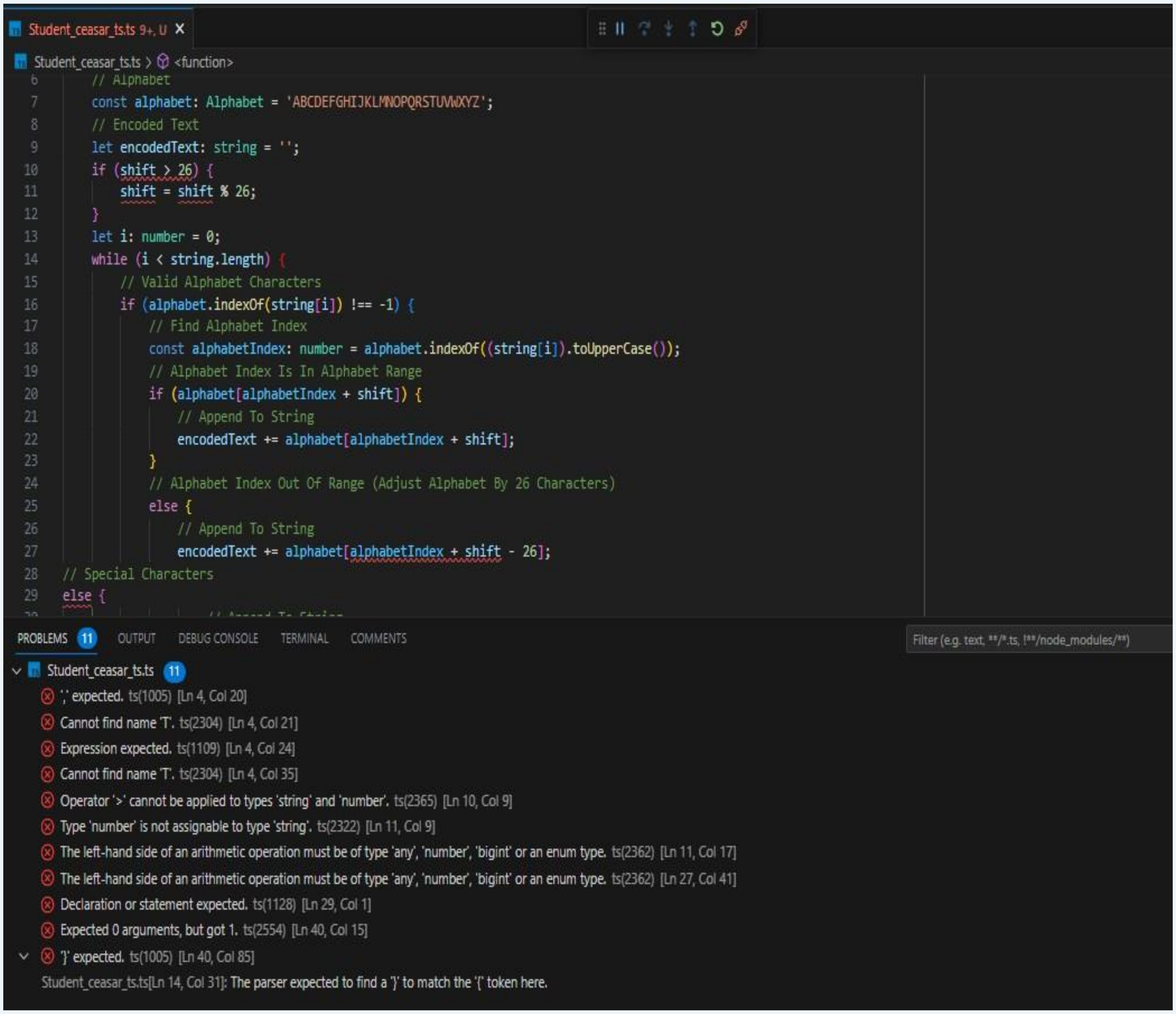
CODE REVIEW RUBIC	GRADE & MARKS 10 Marks Each			CODE LINE	ERRORS / ISSUES TO RESOLVE	MENTOR'S REMARK
Code Correctness:	~Correct file name: ~ Displays Output: ~ Clean Coding: Total Marks (30): Grade:	Yes No Yes => =>	10 0 7 17 C+	4,10,11, 14, 27, 29, 40	Errors due to: ~omission or missing comma at the end of some of the code. ~Several codes with open brackets left unclosed. ~Variable not defined error. ~Type Error due to implicit of generic variable <T>. ~Error in assigning operator to the code.	~Fix and clear all errors raised because of unassigned or wrongly assigned variables. ~Ensure all opened functions are closed with end to avoid memory leak. ~Fix the type error and implicit generic variable error. ~ Fix the error with operators wrongly assigned.
Efficiency:	~Code Reliability: ~ Speed to compile: ~ Methodology: Total Marks (30): Grade:	Fair Excl. Fair => =>	4 10 3 17 C+	4 - 28	~Multiple errors in the code, because of implicit definition of generic variable. ~Missing Code end commas, missing closing brackets to open coding brackets. ~Type error due to incorrect assignment of variable type and operators.	~It is good practice to always have a main () method which is an entry point and display the output of your program resulting in a neater, and the input codes in modules such as functions, methods this helps enhance the code and result in a neater organized reusable code. ~It is good practice to let your program accept user input, which is then processed, and the compiler displays the desired output because it enhances user experience.
Style:	~Indentation: ~Alignment: ~Name Convention: Total Marks (30): Grade:	Yes Yes Excl => =>	10 10 10 30 A	1-29	To Learn more, read this: Typescript Documentation[ix].	~It is good practice to let Class name match the file name. It helps you or any programmer assigned to maintain the codes to be able to identify which class is called in case your code contains more than one class. ~Avoid the use of implicit variables, it may cause unwanted memory leak, compile error or runtime errors.
Documentation	~Comments: ~Documentation: Total Marks (10): Grade:	Yes Yes => =>	5 3 8 A	1, 6-10, 16-17, 21-22	~Improve on Documentation.	~It is good practice to include comments and documentation in a program it enhances the code to be more comprehensive, easy to understand what each code does and makes code debugging, improvement easier.
Possible Improvement to Enhance Code	-Fix and clear the multiple errors, type errors, generic error and operator assignment error, closing all unclosed brackets. -It is good practice to break your codes into simple functional set of code chunks or modularized with methods and functions, makes it neater, readable and helps make debugging easy. -Ensure to call necessary parameters, explicit definition of variables and careful use of generic coding when necessary but do not use always. -It is very good practice to create a program that takes input from users, easy to use by users and leaving user instructions on how to proceed when using the program.					

Student's Ceasar's cipher TypeScript Task.

This code has errors because:

1. Type Error raised as a result of using implicit definition of generic variable.

2. Missing or omitted end of code semi-colon (;) and open code brackets is not closed.



```
6        // Alphabet
7        const alphabet: Alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
8        // Encoded Text
9        let encodedText: string = '';
10       if (shift > 26) {
11           shift = shift % 26;
12       }
13       let i: number = 0;
14       while (i < string.length) {
15           // Valid Alphabet Characters
16           if (alphabet.indexOf(string[i]) !== -1) {
17               // Find Alphabet Index
18               const alphabetIndex: number = alphabet.indexOf((string[i]).toUpperCase());
19               // Alphabet Index Is In Alphabet Range
20               if (alphabet[alphabetIndex + shift]) {
21                   // Append To String
22                   encodedText += alphabet[alphabetIndex + shift];
23               }
24               // Alphabet Index Out Of Range (Adjust Alphabet By 26 Characters)
25               else {
26                   // Append To String
27                   encodedText += alphabet[alphabetIndex + shift - 26];
28   // Special Characters
29   else {
```

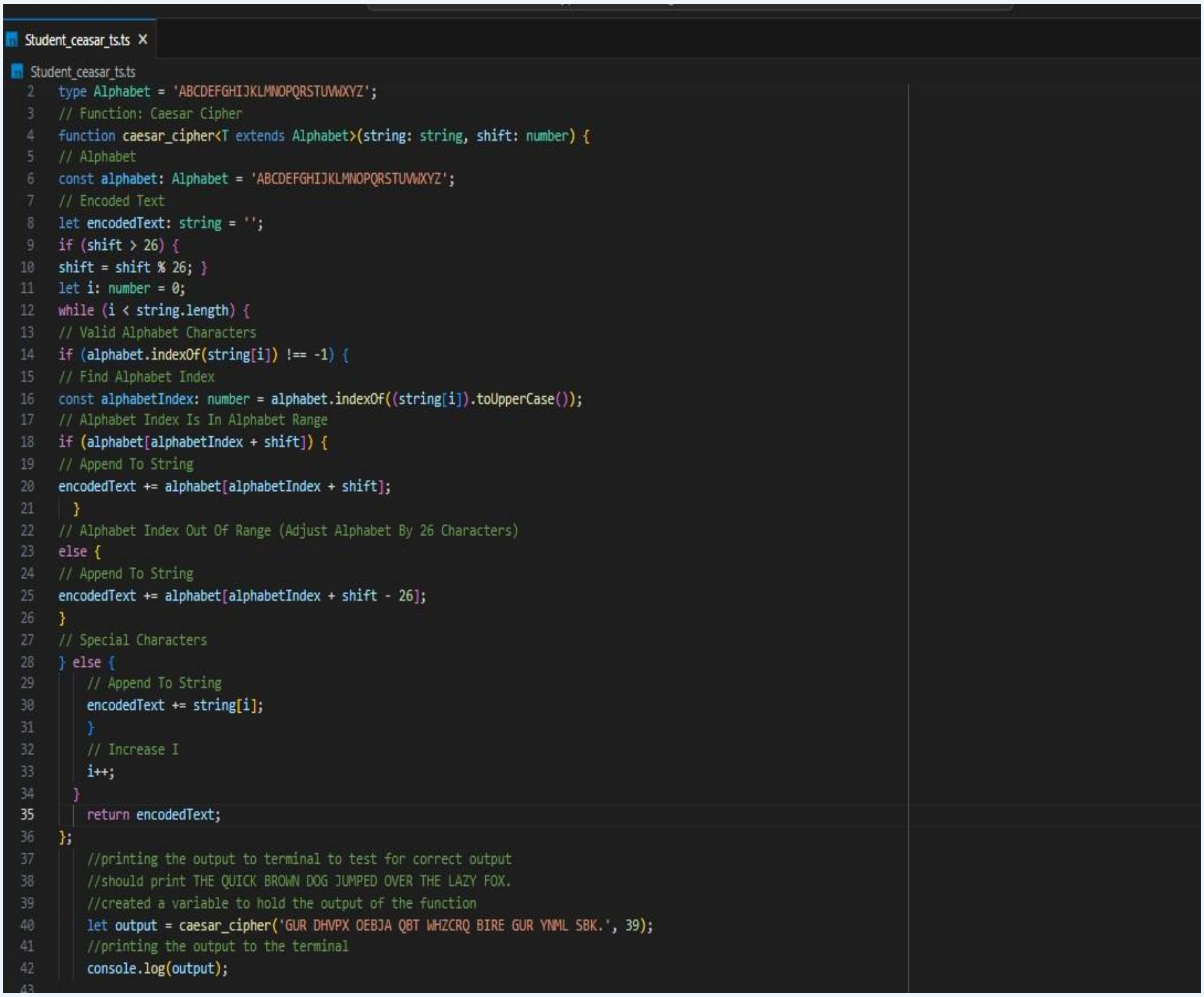
```
Student_ceasar_ts.ts 11
  ';' expected. ts(1005) [Ln 4, Col 20]
  Cannot find name 'T'. ts(2304) [Ln 4, Col 21]
  Expression expected. ts(1109) [Ln 4, Col 24]
  Cannot find name 'T'. ts(2304) [Ln 4, Col 35]
  Operator '>' cannot be applied to types 'string' and 'number'. ts(2365) [Ln 10, Col 9]
  Type 'number' is not assignable to type 'string'. ts(2322) [Ln 11, Col 9]
  The left-hand side of an arithmetic operation must be of type 'any', 'number', 'bigint' or an enum type. ts(2362) [Ln 11, Col 17]
  The left-hand side of an arithmetic operation must be of type 'any', 'number', 'bigint' or an enum type. ts(2362) [Ln 27, Col 41]
  Declaration or statement expected. ts(1128) [Ln 29, Col 1]
  Expected 0 arguments, but got 1. ts(2554) [Ln 40, Col 15]
  '}' expected. ts(1005) [Ln 40, Col 85]
    Student_ceasar_ts.ts[Ln 14, Col 31]: The parser expected to find a '}' to match the '{' token here.
```

Steps on how to fix the error in the student’s Caesar’s cipher file:

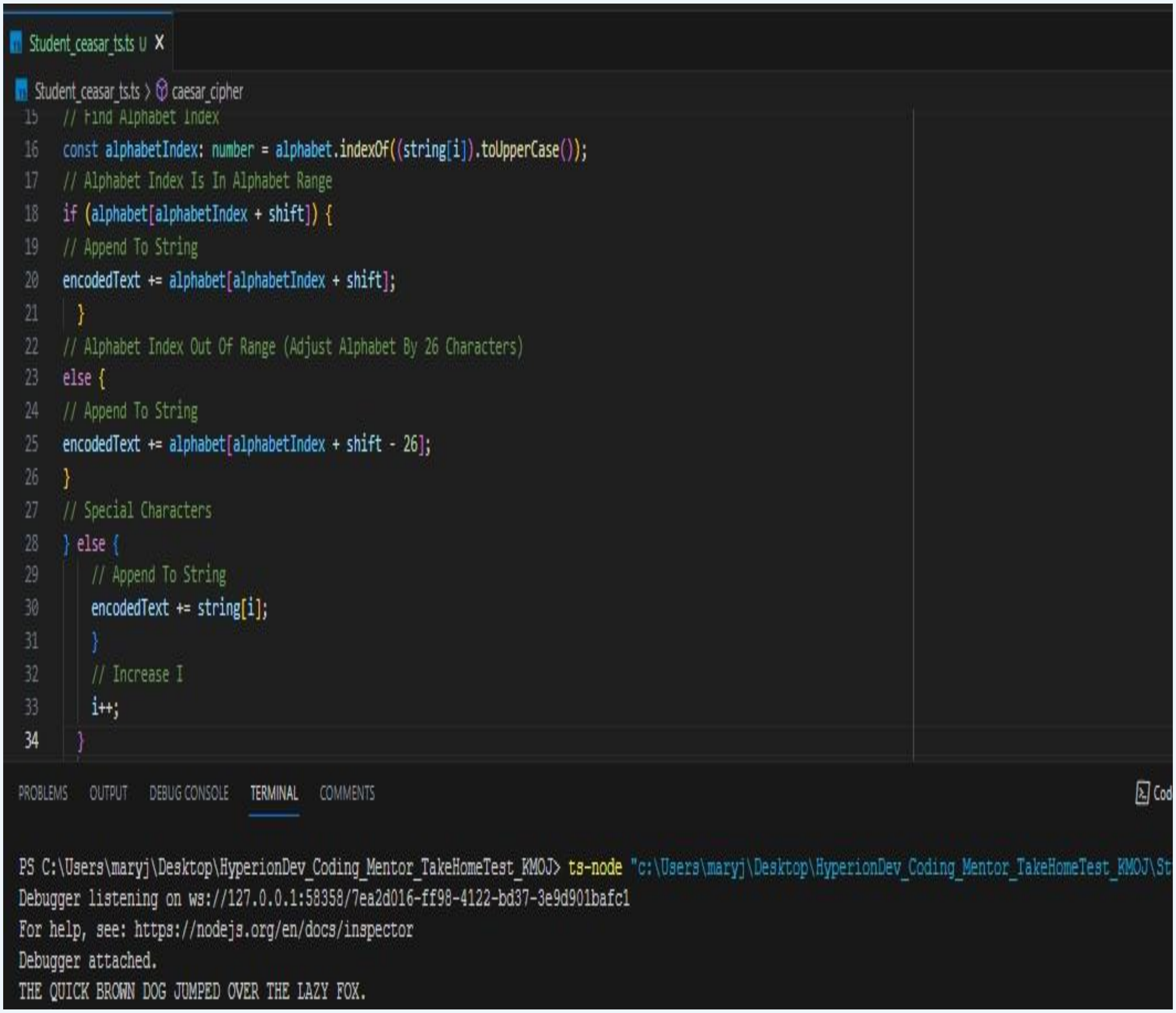
1. Close all the open brackets, end all code statement with a comma.

2. Fix the type error and implicit generic code definition.

3. Add output to test the function and modify the output display to be less cumbersome and make it more meaningful to appeal to the user and increase user experience.



```typescript
type Alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
// Function: Caesar Cipher
function caesar_cipher<T extends Alphabet>(string: string, shift: number) {
// Alphabet
const alphabet: Alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
// Encoded Text
let encodedText: string = '';
if (shift > 26) {
shift = shift % 26; }
let i: number = 0;
while (i < string.length) {
// Valid Alphabet Characters
if (alphabet.indexOf(string[i]) !== -1) {
// Find Alphabet Index
const alphabetIndex: number = alphabet.indexOf((string[i]).toUpperCase());
// Alphabet Index Is In Alphabet Range
if (alphabet[alphabetIndex + shift]) {
// Append To String
encodedText += alphabet[alphabetIndex + shift];
  }
// Alphabet Index Out Of Range (Adjust Alphabet By 26 Characters)
else {
// Append To String
encodedText += alphabet[alphabetIndex + shift - 26];
}
// Special Characters
} else {
    // Append To String
    encodedText += string[i];
    }
    // Increase I
    i++;
  }
    return encodedText;
};
    //printing the output to terminal to test for correct output
    //should print THE QUICK BROWN DOG JUMPED OVER THE LAZY FOX.
    //created a variable to hold the output of the function
    let output = caesar_cipher('GUR DHVPX OEBJA QBT WHZCRQ BIRE GUR YNML SBK.', 39);
    //printing the output to the terminal
    console.log(output);
```



```
// Find Alphabet Index
const alphabetIndex: number = alphabet.indexOf((string[i]).toUpperCase());
// Alphabet Index Is In Alphabet Range
if (alphabet[alphabetIndex + shift]) {
// Append To String
encodedText += alphabet[alphabetIndex + shift];
  }
// Alphabet Index Out Of Range (Adjust Alphabet By 26 Characters)
else {
// Append To String
encodedText += alphabet[alphabetIndex + shift - 26];
}
// Special Characters
} else {
    // Append To String
    encodedText += string[i];
    }
    // Increase I
    i++;
  }
```

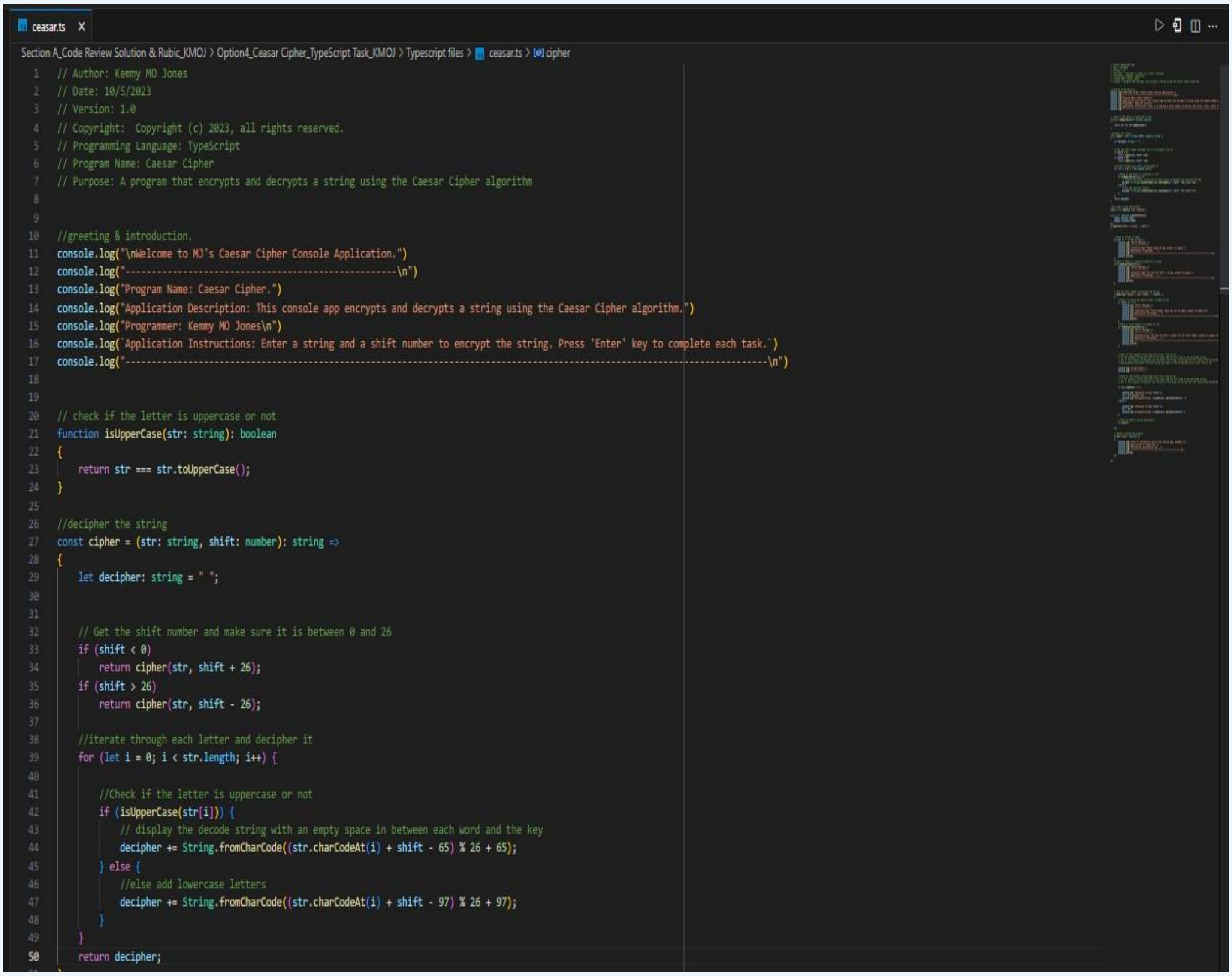
```
PS C:\Users\maryj\Desktop\HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ> ts-node "c:\Users\maryj\Desktop\HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ\St
Debugger listening on ws://127.0.0.1:58358/7ea2d016-ff98-4122-bd37-3e9d901bafc1
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.
THE QUICK BROWN DOG JUMPED OVER THE LAZY FOX.
```

The above shows the student’s Ceasar's cipher program output after all the above stated errors were cleared and fixed.

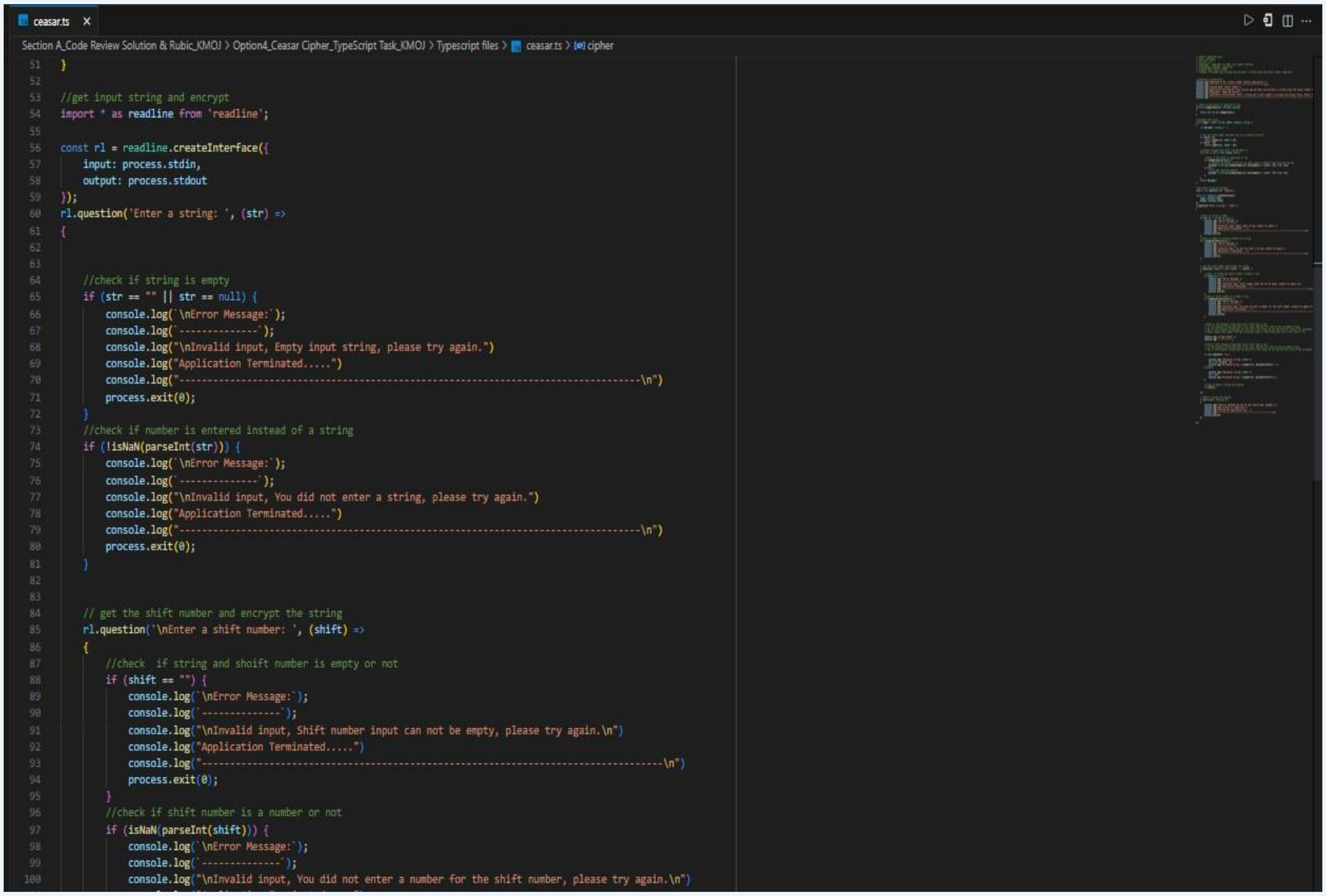
MENTOR’s EXAMPLE SOLUTION (INPUT)

Caesar’s cipher file: improvements students can make on their program

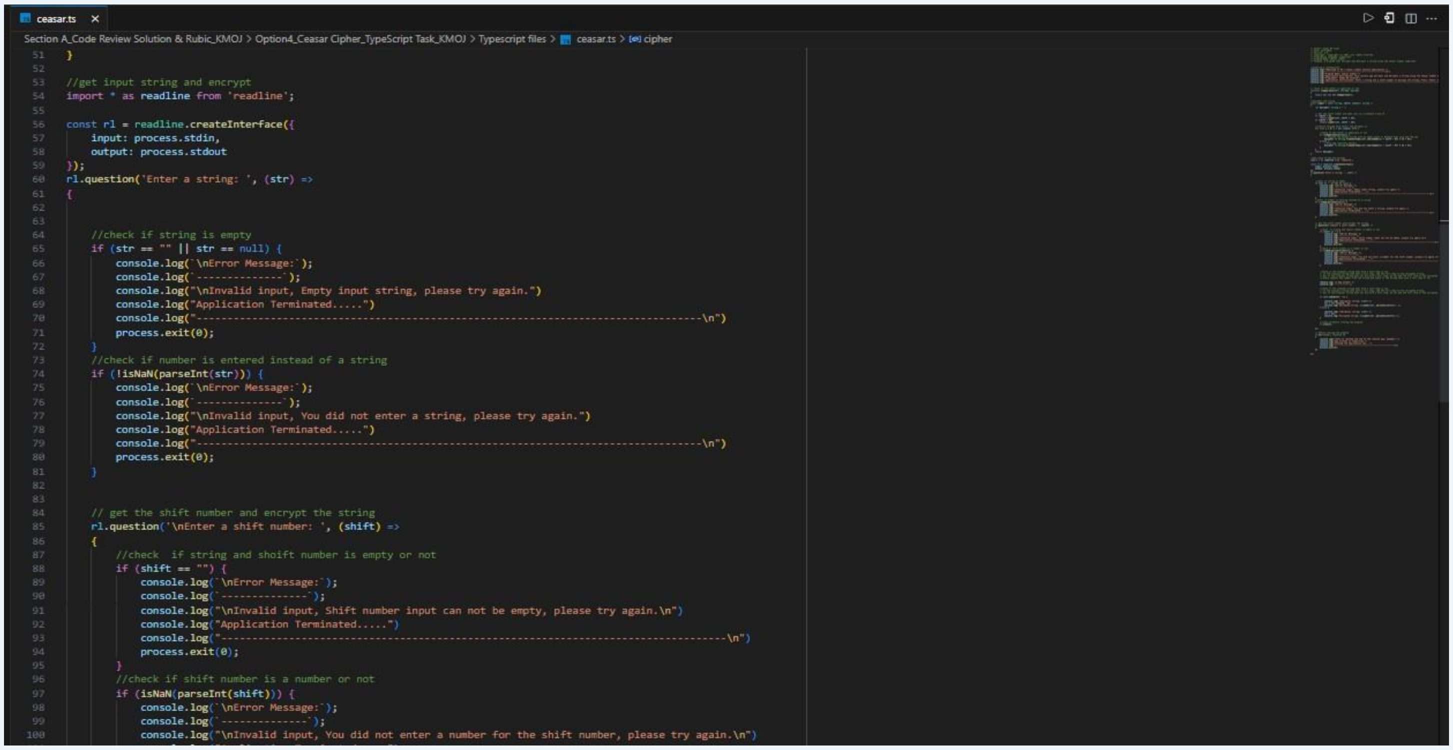
1. **Prompt user for input and displays the anagram of the input string by using the scanner class.**
2. **Code is compartmentalized and modularized, has main method as the entry point, code validation and error handling for example imputing empty string, etc.**
3. **Add documentation and comments on important lines or chunk of code describing what the code does.**
4. **Add instruction and feedback to user on the task success or completion and error message display in**


ceasar.ts
Section A_Code Review Solution & Rubic_KMOJ > Option4_Ceasar Cipher_TypeScript Task_KMOJ > Typescript files > ceasar.ts > cipher

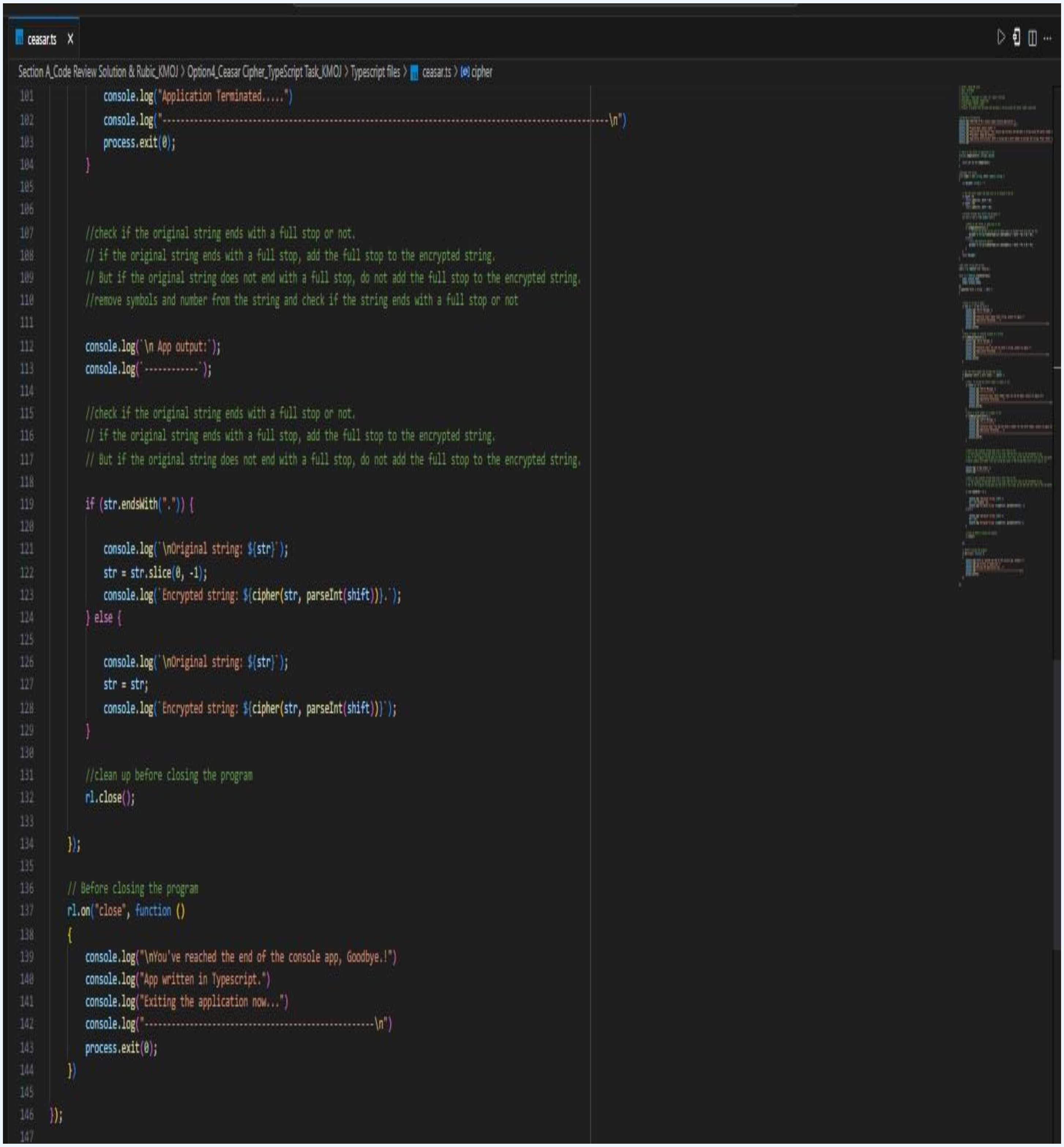
```
// Author: Kemmy MO Jones
// Date: 10/5/2023
// Version: 1.0
// Copyright:  Copyright (c) 2023, all rights reserved.
// Programming Language: TypeScript
// Program Name: Caesar Cipher
// Purpose: A program that encrypts and decrypts a string using the Caesar Cipher algorithm


//greeting & introduction.
console.log("\nWelcome to MJ's Caesar Cipher Console Application.")
console.log("---------------------------------------------------\n")
console.log("Program Name: Caesar Cipher.")
console.log("Application Description: This console app encrypts and decrypts a string using the Caesar Cipher algorithm.")
console.log("Programmer: Kemmy MO Jones\n")
console.log(`Application Instructions: Enter a string and a shift number to encrypt the string. Press 'Enter' key to complete each task.`)
console.log("--------------------------------------------------------------------------------------------------------------------------\n")


// check if the letter is uppercase or not
function isUpperCase(str: string): boolean
{
    return str === str.toUpperCase();
}

//decipher the string
const cipher = (str: string, shift: number): string =>
{
    let decipher: string = " ";


    // Get the shift number and make sure it is between 0 and 26
    if (shift < 0)
        return cipher(str, shift + 26);
    if (shift > 26)
        return cipher(str, shift - 26);

    //iterate through each letter and decipher it
    for (let i = 0; i < str.length; i++) {

        //Check if the letter is uppercase or not
        if (isUpperCase(str[i])) {
            // display the decode string with an empty space in between each word and the key
            decipher += String.fromCharCode((str.charCodeAt(i) + shift - 65) % 26 + 65);
        } else {
            //else add lowercase letters
            decipher += String.fromCharCode((str.charCodeAt(i) + shift - 97) % 26 + 97);
        }
    }
    return decipher;
```


ceasar.ts
Section A_Code Review Solution & Rubic_KMOJ > Option4_Ceasar Cipher_TypeScript Task_KMOJ > Typescript files > ceasar.ts > cipher

```
}

//get input string and encrypt
import * as readline from 'readline';

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});
rl.question('Enter a string: ', (str) =>
{


    //check if string is empty
    if (str == "" || str == null) {
        console.log(`\nError Message:`);
        console.log(`-------------`);
        console.log("\nInvalid input, Empty input string, please try again.")
        console.log("Application Terminated.....")
        console.log("-------------------------------------------------------------------------------\n")
        process.exit(0);
    }
    //check if number is entered instead of a string
    if (!isNaN(parseInt(str))) {
        console.log(`\nError Message:`);
        console.log(`-------------`);
        console.log("\nInvalid input, You did not enter a string, please try again.")
        console.log("Application Terminated.....")
        console.log("-------------------------------------------------------------------------------\n")
        process.exit(0);
    }


    // get the shift number and encrypt the string
    rl.question('\nEnter a shift number: ', (shift) =>
    {
        //check  if string and shoift number is empty or not
        if (shift == "") {
            console.log(`\nError Message:`);
            console.log(`-------------`);
            console.log("\nInvalid input, Shift number input can not be empty, please try again.\n")
            console.log("Application Terminated.....")
            console.log("-------------------------------------------------------------------------------\n")
            process.exit(0);
        }
        //check if shift number is a number or not
        if (isNaN(parseInt(shift))) {
            console.log(`\nError Message:`);
            console.log(`-------------`);
            console.log("\nInvalid input, You did not enter a number for the shift number, please try again.\n")
```



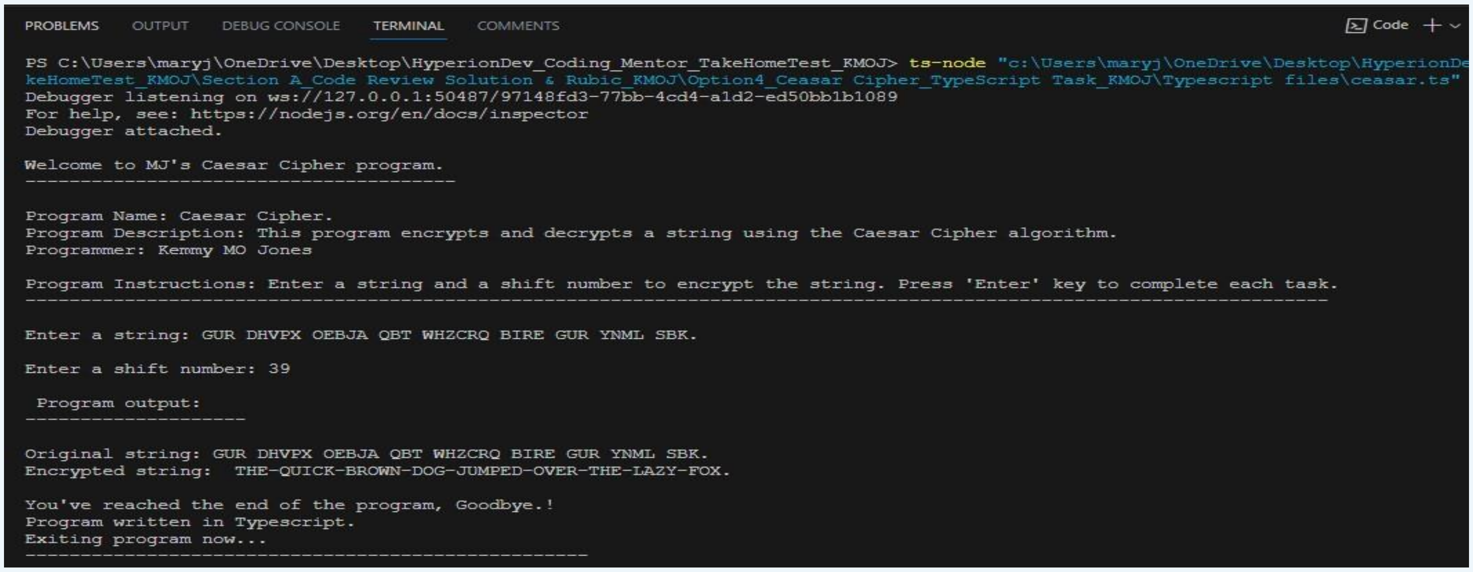
```
}

//get input string and encrypt
import * as readline from 'readline';

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});
rl.question('Enter a string: ', (str) =>
{


    //check if string is empty
    if (str == "" || str == null) {
        console.log(`\nError Message:`);
        console.log(`-------------`);
        console.log("\nInvalid input, Empty input string, please try again.")
        console.log("Application Terminated.....")
        console.log("-------------------------------------------------------------------------------\n")
        process.exit(0);
    }
    //check if number is entered instead of a string
    if (!isNaN(parseInt(str))) {
        console.log(`\nError Message:`);
        console.log(`-------------`);
        console.log("\nInvalid input, You did not enter a string, please try again.")
        console.log("Application Terminated.....")
        console.log("-------------------------------------------------------------------------------\n")
        process.exit(0);
    }


    // get the shift number and encrypt the string
    rl.question('\nEnter a shift number: ', (shift) =>
    {
        //check  if string and shoift number is empty or not
        if (shift == "") {
            console.log(`\nError Message:`);
            console.log(`-------------`);
            console.log("\nInvalid input, Shift number input can not be empty, please try again.\n")
            console.log("Application Terminated.....")
            console.log("-------------------------------------------------------------------------------\n")
            process.exit(0);
        }
        //check if shift number is a number or not
        if (isNaN(parseInt(shift))) {
            console.log(`\nError Message:`);
            console.log(`-------------`);
            console.log("\nInvalid input, You did not enter a number for the shift number, please try again.\n")
```


ceasar.ts X
Section A_Code Review Solution & Rubic_KMOJ > Option4_Ceasar Cipher_TypeScript Task_KMOJ > Typescript files > ceasar.ts > cipher

```
                console.log("Application Terminated.....")
                console.log("----------------------------------------------------------------------------------------------------\n")
                process.exit(0);
            }


            //check if the original string ends with a full stop or not.
            // if the original string ends with a full stop, add the full stop to the encrypted string.
            // But if the original string does not end with a full stop, do not add the full stop to the encrypted string.
            //remove symbols and number from the string and check if the string ends with a full stop or not

            console.log(`\n App output:`);
            console.log(`-----------`);

            //check if the original string ends with a full stop or not.
            // if the original string ends with a full stop, add the full stop to the encrypted string.
            // But if the original string does not end with a full stop, do not add the full stop to the encrypted string.

            if (str.endsWith(".")) {

                console.log(`\nOriginal string: ${str}`);
                str = str.slice(0, -1);
                console.log(`Encrypted string: ${cipher(str, parseInt(shift))}.`);
            } else {

                console.log(`\nOriginal string: ${str}`);
                str = str;
                console.log(`Encrypted string: ${cipher(str, parseInt(shift))}`);
            }

            //clean up before closing the program
            rl.close();

        });

        // Before closing the program
        rl.on("close", function ()
        {
            console.log("\nYou've reached the end of the console app, Goodbye.!")
            console.log("App written in Typescript.")
            console.log("Exiting the application now...")
            console.log("----------------------------------------------------\n")
            process.exit(0);
        })

    });
```

MENTOR's EXAMPLE SOLUTION (OUTPUT)

Caesar.ts file: The Caesar's cipher console app output is shown below



```
PS C:\Users\maryj\OneDrive\Desktop\HyperionDev_Coding_Mentor_TakeHomeTest_KMOJ> ts-node "c:\Users\maryj\OneDrive\Desktop\HyperionDe
keHomeTest_KMOJ\Section A_Code Review Solution & Rubic_KMOJ\Option4_Ceasar Cipher_TypeScript Task_KMOJ\Typescript files\ceasar.ts"
Debugger listening on ws://127.0.0.1:50487/97148fd3-77bb-4cd4-a1d2-ed50bb1b1089
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.

Welcome to MJ's Caesar Cipher program.
---------------------------------------

Program Name: Caesar Cipher.
Program Description: This program encrypts and decrypts a string using the Caesar Cipher algorithm.
Programmer: Kemmy MO Jones

Program Instructions: Enter a string and a shift number to encrypt the string. Press 'Enter' key to complete each task.
--------------------------------------------------------------------------------------------------------------------------

Enter a string: GUR DHVPX OEBJA QBT WHZCRQ BIRE GUR YNML SBK.

Enter a shift number: 39

 Program output:
--------------------

Original string: GUR DHVPX OEBJA QBT WHZCRQ BIRE GUR YNML SBK.
Encrypted string:  THE-QUICK-BROWN-DOG-JUMPED-OVER-THE-LAZY-FOX.

You've reached the end of the program, Goodbye.!
Program written in Typescript.
Exiting program now...
---------------------------------------------------
```

References:

[ii] Download Python Interpreter and install your computer: Download Python Interpreter

- Install code runner extension on visual studio code and set it your python interpreter: Programming Style, Official Python Documentation

[iii] Coding Standards and Best Practices to Follow , Code Like a Pythonista: Idiomatic Python by David Goodger

[iv] Official Java Tutorial (Oracle)

[v.] Download and install latest version of jdk on your computer: Adoptium (Temurin Jdk 11) , Java Jdk (Oracle)

-Configure your visual studio to Run Java after installing any jdk of your choice.

-Install Java Debugger extension in visual studio: Extension Pack for Java, Java Debugger, Java Run

-Any other helpful extension you need to run your codes in java.

[vi] Ruby Conversion Methods

[vii] How To Write Algorithm In Programming by Levine Nicole , Building Blocks of Algorithm By Khan Academy , Pseudocode by Wikipedia

[viii.] Download and install latest version of ruby installer on your computer: Ruby Installer

-Configure your visual studio to Run Ruby codes after installing any Ruby or Ruby Devkit of your choice.

-Install Ruby Debugger extension in visual studio: Ruby by Peng LV

-To Learn More: Pseudocode: A Beginner's Guide by Bello

[ix] TypeScript: JavaScript With Syntax For Types. (typescriptlang.org)