Report

Introduction

The software company aims to develop tools for numerology analysis, focusing on two scenarios:

For a given birthday, I determined the Life Path Number, identified the Lucky Colour, and checked if it's a master number. Also, I compared Life Path Numbers if two birthdays are provided.

After that, I determined the generation a person belongs to based on their birthday. Only birthdays between 1901 and 2024 were considered, accepting input in both numeric and month name formats.

To do that I created **overloaded functions**.

Module Descriptions

public int CheckTwoBirthdays(int date1, int mon1, int year1, int date2, int mon2, int year2)

This method compares two birthdays and returns the Life Path Number of the person with the higher Life Path Number. If the Life Path Numbers are the same, it returns 1 otherwise it returns 0.

Method uses **LifePathNumber** funtion two times so its reusable.

public int CheckTwoBirthdays(int date1, String mon1, int year1, int date2, String mon2, int year2)

This method compares two birthdays same as previous function. Difference is only in function parameters. It is an overloaded function.

public int CheckTwoBirthdays(String birthdate1, String birthdate2)

This method compares two birthdays same as previous function. Difference is only in function parameters. It is an overloaded function.

public String FindGeneration(String birthdate)

This method determines the generation a person belongs to based on their birthday. It returns the generation name.

public String FindGeneration(int date, int mon, int year)

This method determines the generation a person belongs to based on their birthday. It returns the generation name. It is an overloaded function.

public boolean MasterNumber(int lpn)

This method checks if the Life Path Number is a master number. It returns true if it is a master number, otherwise it returns false.

public String LuckyColour(int lpn)

This method returns the lucky colour based on the Life Path Number.

public int LifePathNumber(int date, int month, int year)

This method calculates the Life Path Number based on the birthday. It returns the Life Path Number.

uses **sumLPN** in a loop to calculate LPN

public int LifePathNumber(String birthdate)

This method calculates the Life Path Number based on the birthday. It returns the Life Path Number. It is an overloaded function. parameter is string which is splitted on the basis of '-' to extract day month and year.

public int LifePathNumber(int date, String month, int year)

This method calculates the Life Path Number based on the birthday. It returns the Life Path Number. It is an overloaded function. Month number is calculated through Month() function.

private int sumLPN(int num)

This method calculates the sum of the digits of the Life Path Number until it becomes one digit. It returns the sum.

private int Month(String mon)

This method converts the month name to a number. It returns the month number.

Modularity

The program is divided into modules that perform specific tasks. Each module is responsible for a specific task, and the modules are designed to be reusable and interchangeable.

Modularization:

The class is divided into multiple methods, each responsible for a specific task. For example, methods like CheckTwoBirthdays, FindGeneration, MasterNumber, LuckyColour, and LifePathNumber represent modularization by breaking down the functionality into smaller, manageable units.

Encapsulation:

Encapsulation is evident in the way methods encapsulate related functionalities. For example, CheckTwoBirthdays encapsulates logic related to comparing two birthdays, FindGeneration determines the generation based on birthdate, MasterNumber checks for master numbers, and so on. Each method hides its implementation details and provides a clear interface to interact with.

Abstraction:

Abstraction is demonstrated through the method signatures and their usage. Users interact with the class by calling methods like CheckTwoBirthdays, FindGeneration, etc., without needing to know the internal workings of these methods. The abstraction allows users to focus on what each method does rather than how it accomplishes it.

Separation of Concerns:

Each method within the class addresses a specific concern or functionality. For instance, CheckTwoBirthdays deals with comparing two birthdays, FindGeneration determines the generation, MasterNumber checks for master numbers, and so on. This separation of concerns makes the codebase easier to understand, maintain, and modify.

Running Program

Run TestCases.java

javac TestCases.java

java TestCases

Now a menu driven interface will be shown. The programs asks whether you want to **Run the TestCases** or **enter input from user** and store it in result.txt or **read data from input.txt** already present in the same folder and display output on terminal. If you want to run the test cases, press 1 and then press enter. If you want to enter input from user, press 2 and if you want to import from input.txt, press 3. Press -1 to end the program.

If choice 2 or 3 is selected then the Life Path Number is calculated using functions defined above which are stored in a separate file named **LifePathCalculator**. The Life Path Number is then used to calculate the Master Number and the Lucky Colour.

input.txt contains last 4 digits of roll no as year

Black Box Test Cases

Testing Functions

EquivalencePartitioning_LifePathNumber(calculator); EquivalencePartitioning_MasterNumber(calculator); EquivalencePartitioning_LuckyColour(calculator); EquivalencePartitioning_Generation(calculator); EquivalencePartitioning_CheckTwoBirthdays(calculator);

BoundaryValue_LifePathNumber(calculator); BoundaryValue_LuckyColour(calculator); BoundaryValue_Generation(calculator);

Test Case	Description	Expected Output	Status	Comments
1	Life Path Number EP(15-June-1990)	4	Pass	Valid
2	Life Path Number EP(00-June-1990)	-1	Pass	0 Day Invalid
3	Life Path Number EP(33-June-1990)	-1	Pass	33 Day Invalid
4	Life Path Number EP(15-June-1990)	4	Pass	Valid
5	Life Path Number EP(15-June-1990)	4	Pass	Valid
6	Life Path Number EP(15-June-1800)	-1	Pass	year < 1901 Invalid

Test Case	Description	Expected Output	Status	Comments
7	Life Path Number EP(15-June-2080)	-1	Pass	year>2024 Invalid
8	Life Path Number EP(15-June-1990)	4	Pass	Valid
9	Master Number EP1(11)	true	Pass	Valid
10	Master Number EP2(1)	false	Pass	Valid
11	Master Number EP3(0)	false	Pass	Valid
12	Master Number EP4(34)	false	Pass	Valid
13	Lucky Colour EP5(1)	"Red"	Pass	Valid
14	Lucky Colour EP6(5)	"Sky Blue"	Pass	Valid
15	Lucky Colour EP7(9)	"Gold"	Pass	Valid
16	Lucky Colour EP8(11)	"Silver"	Pass	Valid
17	Lucky Colour EP9(22)	"White"	Pass	Valid
18	Lucky Colour EP10(33)	"Crimson"	Pass	Valid
19	Lucky Colour EP11(0)	""	Pass	0<1 InValid
20	Lucky Colour EP12(34)	""	Pass	34>33 InValid
21	Generation EP13(15-01-1945)	"Silent"	Pass	Valid
22	Generation EP14(15-01-1955)	"Baby Boomers"	Pass	Valid
23	Generation EP15(15-01-1970)	"Generation X"	Pass	Valid
24	Generation EP16(15-01-1990)	"Millennials"	Pass	Valid
25	Generation EP17(15-01-2005)	"Generation Z"	Pass	Valid
26	Generation EP18(15-01-2015)	"Generation Alpha"	Pass	Valid
27	Generation EP19(15-01-1800)	""	Pass	1800<1901 InValid
28	Generation EP20(15-01-2030)	""	Pass	2030>2024 InValid
29	CheckTwoBirthdays EP21(15-01-2000,15-01-2000)	1	Pass	Valid
30	CheckTwoBirthdays EP22(15-01-1800,20-02-1995)	0	Pass	Valid

Test Case	Description	Expected Output	Status	Comments
31	CheckTwoBirthdays EP23(15-01-1800,120-02-1995)	-1	Pass	1800<1901 InValid
32	CheckTwoBirthdays EP24(15-01-2000,20-02-2100)	-1	Pass	2100>2024 InValid
33	Life Path Number BV(01-01-1901)	4	Pass	Valid
34	Life Path Number BV(00-01-1901)	-1	Pass	Day<1 InValid
35	Life Path Number BV(31-12-2024)	6	Pass	Valid
36	Life Path Number BV(32-12-2024)	-1	Pass	Day>31 InValid
37	Life Path Number BV(15-01-2000)	9	Pass	Valid
38	Life Path Number BV(15-00-2000)	-1	Pass	Month < 1 InValid
39	Life Path Number BV(15-12-2000)	11	Pass	Valid
40	Life Path Number BV(15-13-2000)	-1	Pass	Month>12 InValid
40	Life Path Number BV(15-06-1901)	5	Pass	Valid
41	Life Path Number BV(15-06-1900)	-1	Pass	year<1901 InValid
42	Life Path Number BV(15-06-2024)	2	Pass	Valid
43	Life Path Number BV(15-06-2025)	-1	Pass	year>2024 InValid
44	Lucky Colour BV1(1)	"Red"	Pass	Valid
45	Lucky Colour BV2(5)	"Sky Blue"	Pass	Valid
46	Lucky Colour BV3(9)	"Gold"	Pass	Valid
47	Lucky Colour BV4(11)	"Silver"	Pass	Valid
48	Lucky Colour BV5(22)	"White"	Pass	Valid
49	Lucky Colour BV6(33)	"Crimson"	Pass	Valid
50	Lucky Colour BV7(0)	""	Pass	InValid
51	Lucky Colour BV8(34)	""	Pass	InValid
52	Generation BV9(15-01-1901)	"Silent"	Pass	Valid
53	Generation BV10(15-01-2024)	"Generation Alpha"	Pass	Valid

Test Case	Description	Expected Output	Status	Comments
54	Generation BV11(15-01-1900)	""	Pass	InValid
55	Generation BV12(15-01-2025)	""	Pass	InValid

White Box Test Cases

The requirement was to use white box testing in any two of the modules. White-box testing involves testing internal structures or workings of an application.

CheckTwoBirthdays(int date1, int mon1, int year1, int date2, int mon2, int year2)

Reason for Selection:

Multiple conditional checks Involves calling another method (LifePathNumber) Includes decision-making logic

LifePathNumber(int date, int month, int year)

Reason for Selection:

Multiple conditional checks Internal calculations involving other methods (e.g., sumLPN) Key logic for calculating the life path number

Test Case	Description	Expected Output	Status	Comments
1	CheckTwoBirthdays WB1(1, 1, 1900, 1, 1, 2000)	-1	Pass	1900<1901 InValid
2	CheckTwoBirthdays WB2(1, 1, 2000, 1, 1, 2025)	-1	Pass	2025>2024 Invalid
3	CheckTwoBirthdays WB3(1, 2, 2000, 2, 1, 2000)	1	Pass	Valid
4	CheckTwoBirthdays WB4(1, 1, 2000, 3, 3, 2000)	0	Pass	Valid
5	LifePathNumber WB5(1, 1, 1900)	-1	Pass	year<1901 Invalid
6	LifePathNumber WB6(1, 1, 2025)	-1	Pass	year>2024 Invalid
7	LifePathNumber WB7(0, 1, 2000)	-1	Pass	Day<1 InValid
8	LifePathNumber WB8(32, 1, 2000)	-1	Pass	Day>31 InValid
9	LifePathNumber WB9(15, 6, 1990)	1	Pass	Valid

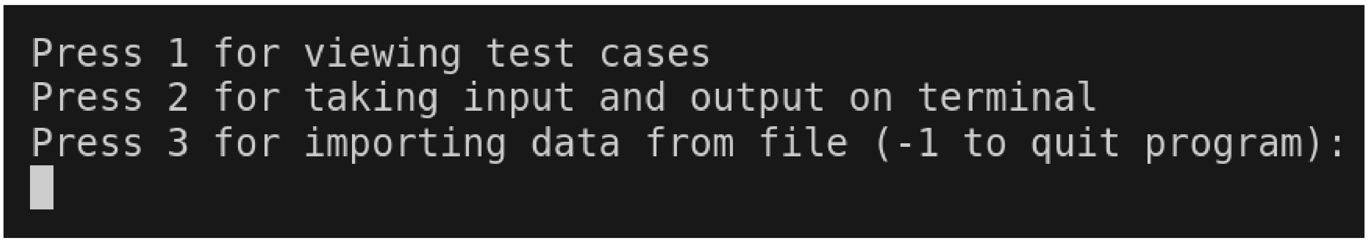
Testing Functions

WhiteBoxForCheckTwoBirthdays(calculator); WhiteBoxForLifePathNumber(calculator);

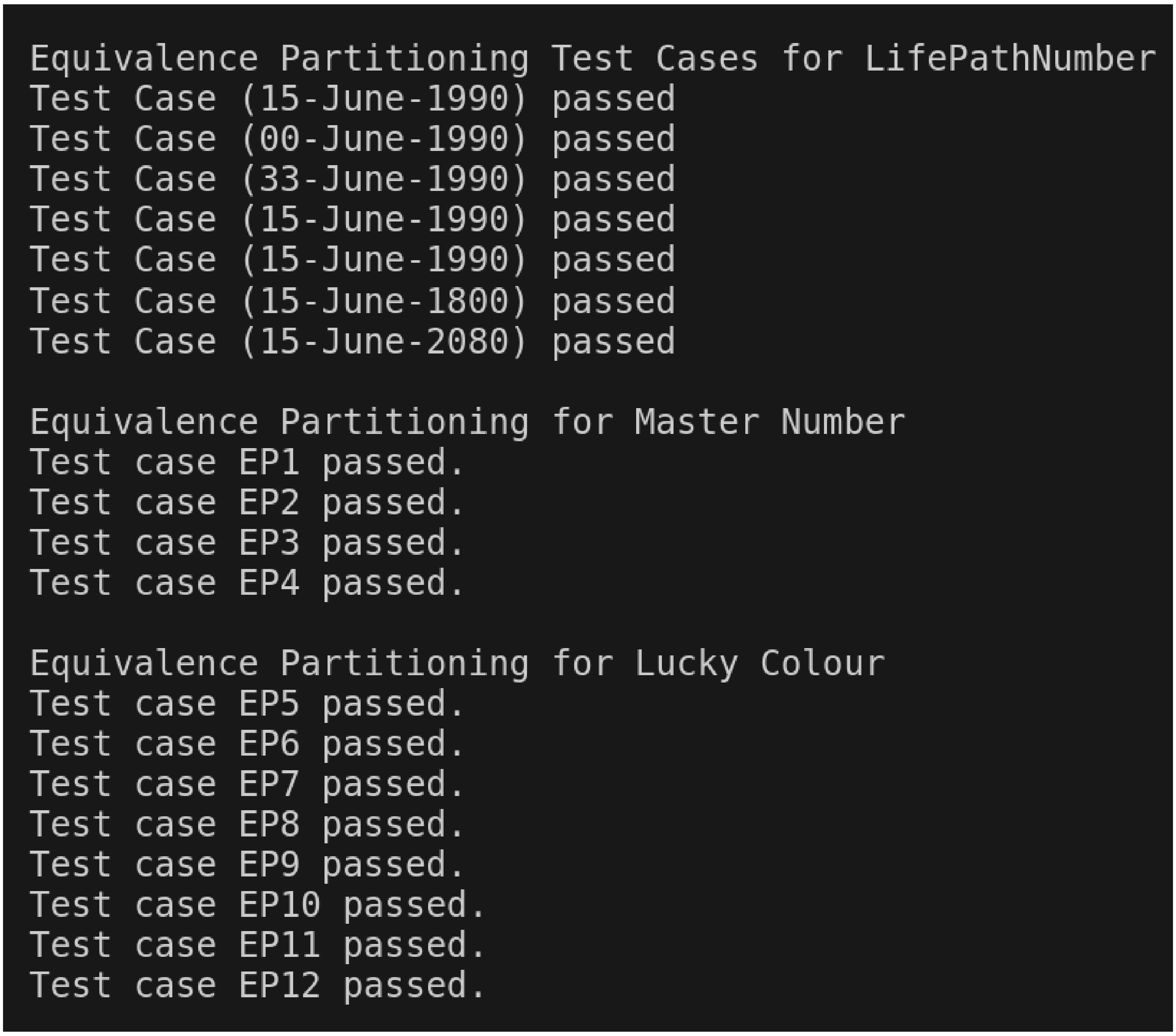
Test implementation and test execution

The test cases were implemented in the file **TestCases.java** and were executed using the command **java TestCases**.

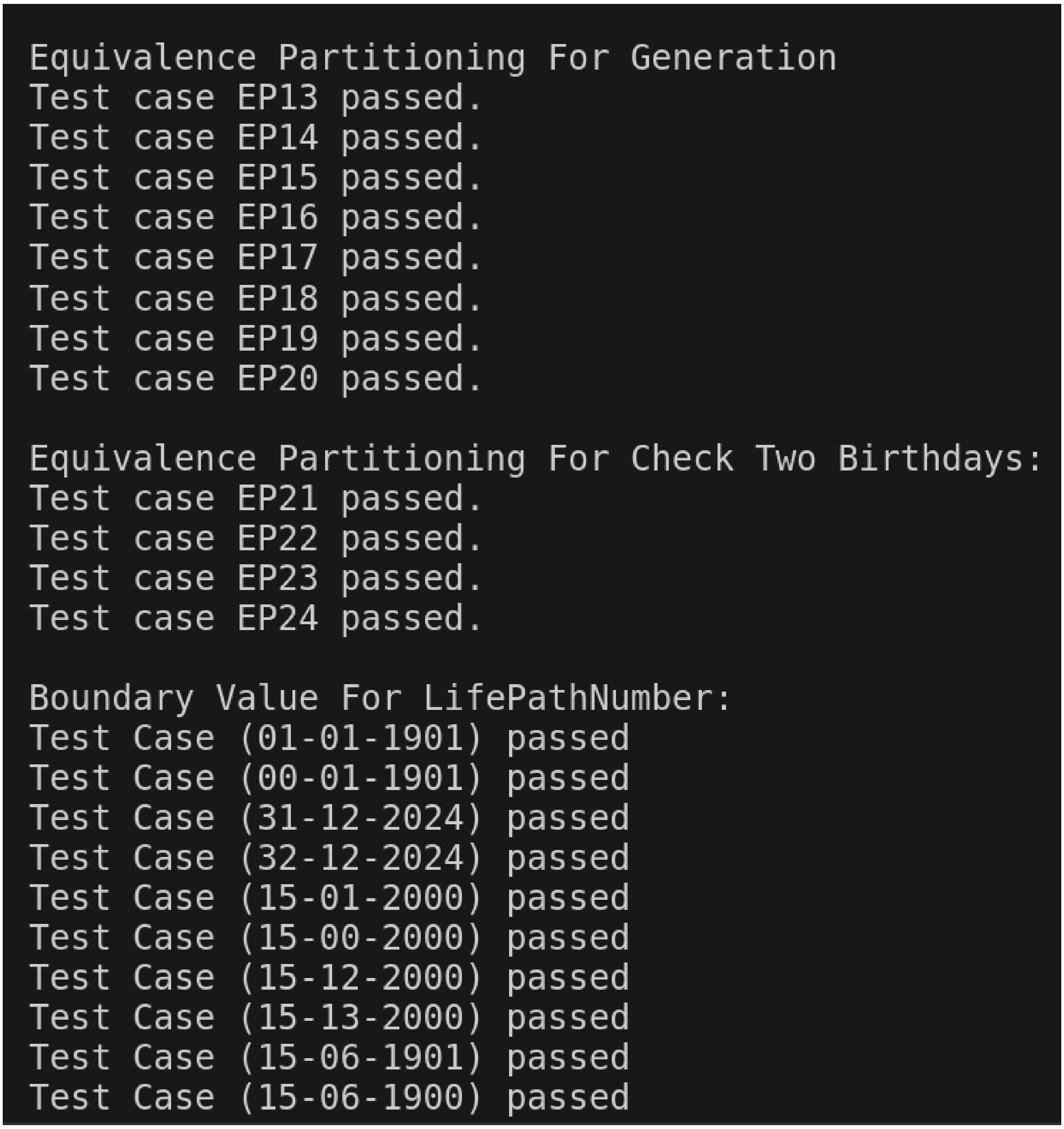
Press 1 to display test case results All test cases are passed Screenshots are attached below



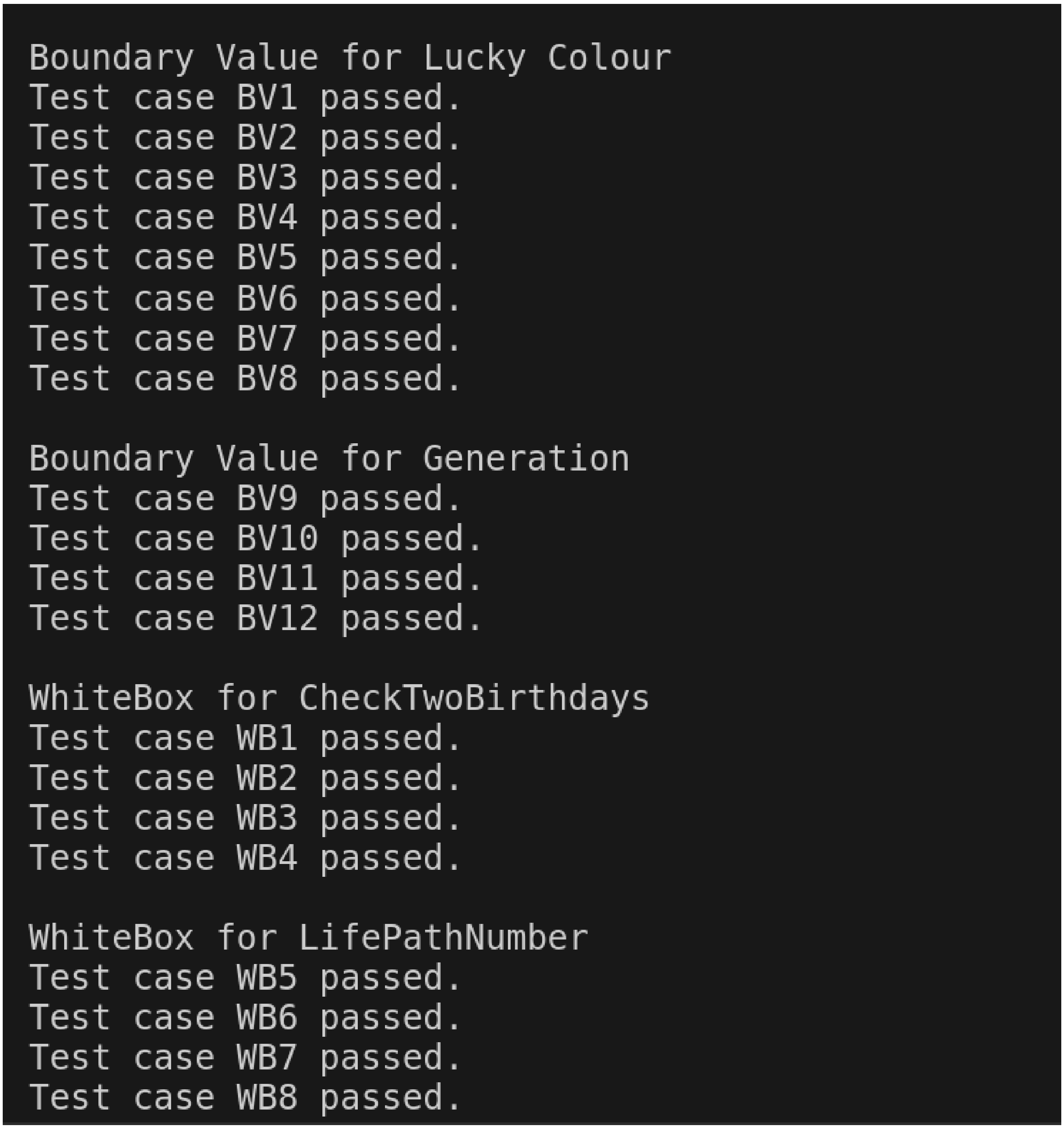
```
Press 1 for viewing test cases
Press 2 for taking input and output on terminal
Press 3 for importing data from file (-1 to quit program):
```



```
Equivalence Partitioning Test Cases for LifePathNumber
Test Case (15-June-1990) passed
Test Case (00-June-1990) passed
Test Case (33-June-1990) passed
Test Case (15-June-1990) passed
Test Case (15-June-1990) passed
Test Case (15-June-1800) passed
Test Case (15-June-2080) passed

Equivalence Partitioning for Master Number
Test case EP1 passed.
Test case EP2 passed.
Test case EP3 passed.
Test case EP4 passed.

Equivalence Partitioning for Lucky Colour
Test case EP5 passed.
Test case EP6 passed.
Test case EP7 passed.
Test case EP8 passed.
Test case EP9 passed.
Test case EP10 passed.
Test case EP11 passed.
Test case EP12 passed.
```

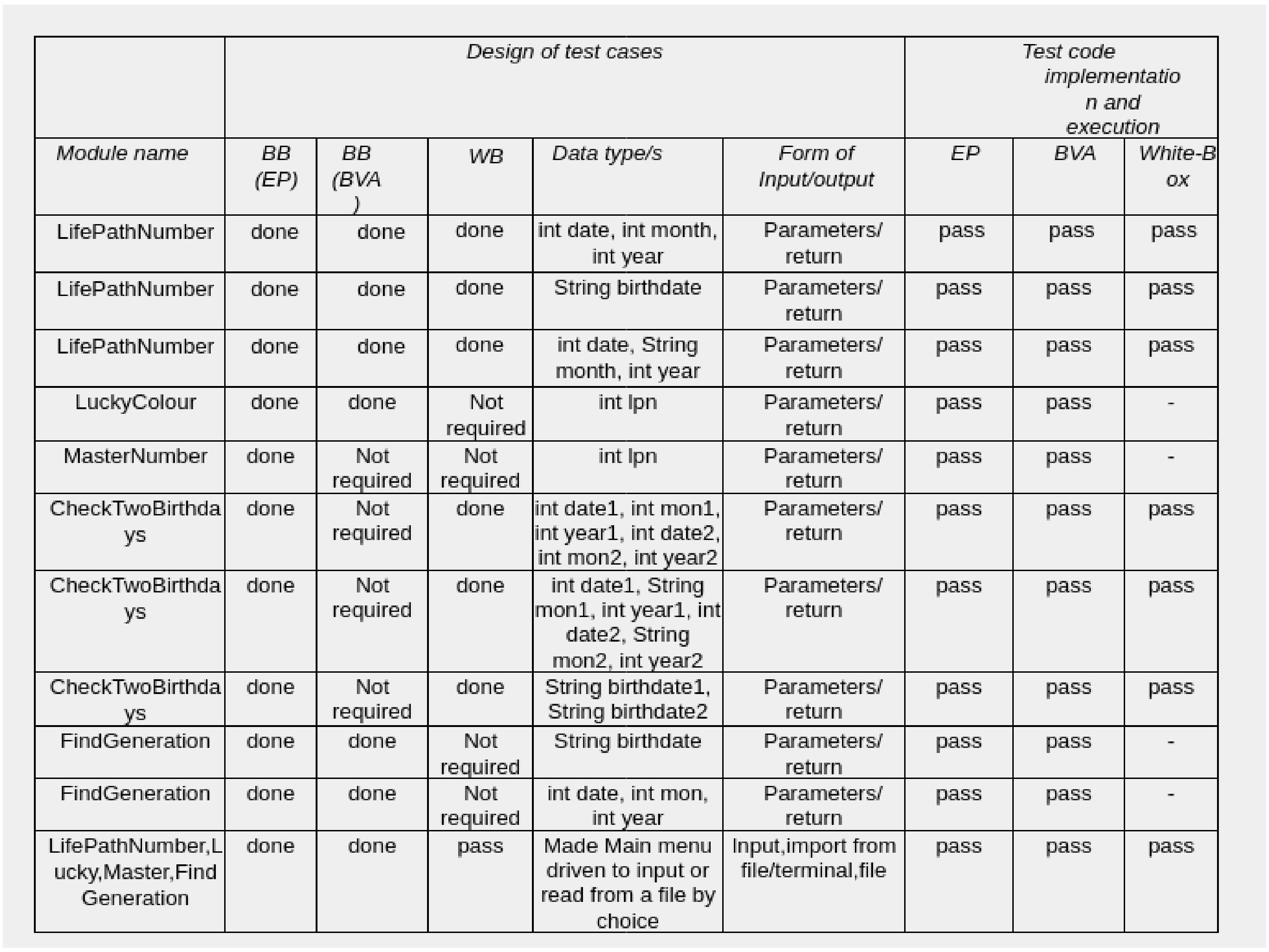


```
Equivalence Partitioning For Generation
Test case EP13 passed.
Test case EP14 passed.
Test case EP15 passed.
Test case EP16 passed.
Test case EP17 passed.
Test case EP18 passed.
Test case EP19 passed.
Test case EP20 passed.

Equivalence Partitioning For Check Two Birthdays:
Test case EP21 passed.
Test case EP22 passed.
Test case EP23 passed.
Test case EP24 passed.

Boundary Value For LifePathNumber:
Test Case (01-01-1901) passed
Test Case (00-01-1901) passed
Test Case (31-12-2024) passed
Test Case (32-12-2024) passed
Test Case (15-01-2000) passed
Test Case (15-00-2000) passed
Test Case (15-12-2000) passed
Test Case (15-13-2000) passed
Test Case (15-06-1901) passed
Test Case (15-06-1900) passed
```



```
Boundary Value for Lucky Colour
Test case BV1 passed.
Test case BV2 passed.
Test case BV3 passed.
Test case BV4 passed.
Test case BV5 passed.
Test case BV6 passed.
Test case BV7 passed.
Test case BV8 passed.

Boundary Value for Generation
Test case BV9 passed.
Test case BV10 passed.
Test case BV11 passed.
Test case BV12 passed.

WhiteBox for CheckTwoBirthdays
Test case WB1 passed.
Test case WB2 passed.
Test case WB3 passed.
Test case WB4 passed.

WhiteBox for LifePathNumber
Test case WB5 passed.
Test case WB6 passed.
Test case WB7 passed.
Test case WB8 passed.
```

Traceability Matrix



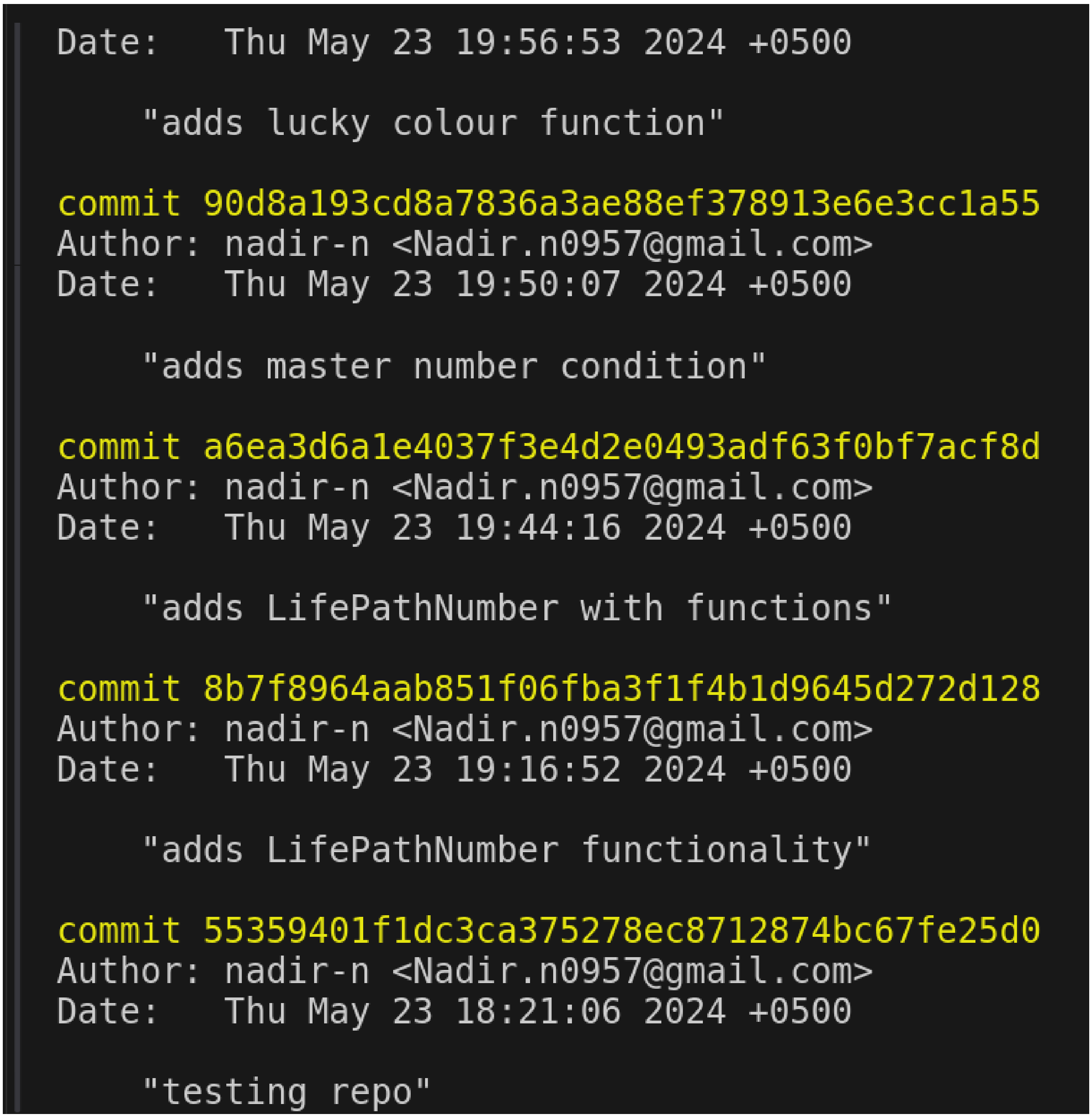
	Design of test cases					*Test code implementation and execution*		
Module name	*BB (EP)*	*BB (BVA)*	*WB*	*Data type/s*	*Form of Input/output*	*EP*	*BVA*	*White-Box*
LifePathNumber	done	done	done	int date, int month, int year	Parameters/ return	pass	pass	pass
LifePathNumber	done	done	done	String birthdate	Parameters/ return	pass	pass	pass
LifePathNumber	done	done	done	int date, String month, int year	Parameters/ return	pass	pass	pass
LuckyColour	done	done	Not required	int lpn	Parameters/ return	pass	pass	-
MasterNumber	done	Not required	Not required	int lpn	Parameters/ return	pass	pass	-
CheckTwoBirthdays	done	Not required	done	int date1, int mon1, int year1, int date2, int mon2, int year2	Parameters/ return	pass	pass	pass
CheckTwoBirthdays	done	Not required	done	int date1, String mon1, int year1, int date2, String mon2, int year2	Parameters/ return	pass	pass	pass
CheckTwoBirthdays	done	Not required	done	String birthdate1, String birthdate2	Parameters/ return	pass	pass	pass
FindGeneration	done	done	Not required	String birthdate	Parameters/ return	pass	pass	-
FindGeneration	done	done	Not required	int date, int mon, int year	Parameters/ return	pass	pass	-
LifePathNumber,Lucky,Master,Find Generation	done	done	pass	Made Main menu driven to input or read from a file by choice	Input,import from file/terminal,file	pass	pass	pass

Text files are used in TestCases.java file to take input in the menu driven part. if 3 is pressed when the program starts, It takes input from a file named input.txt The output is displayed on **terminal**

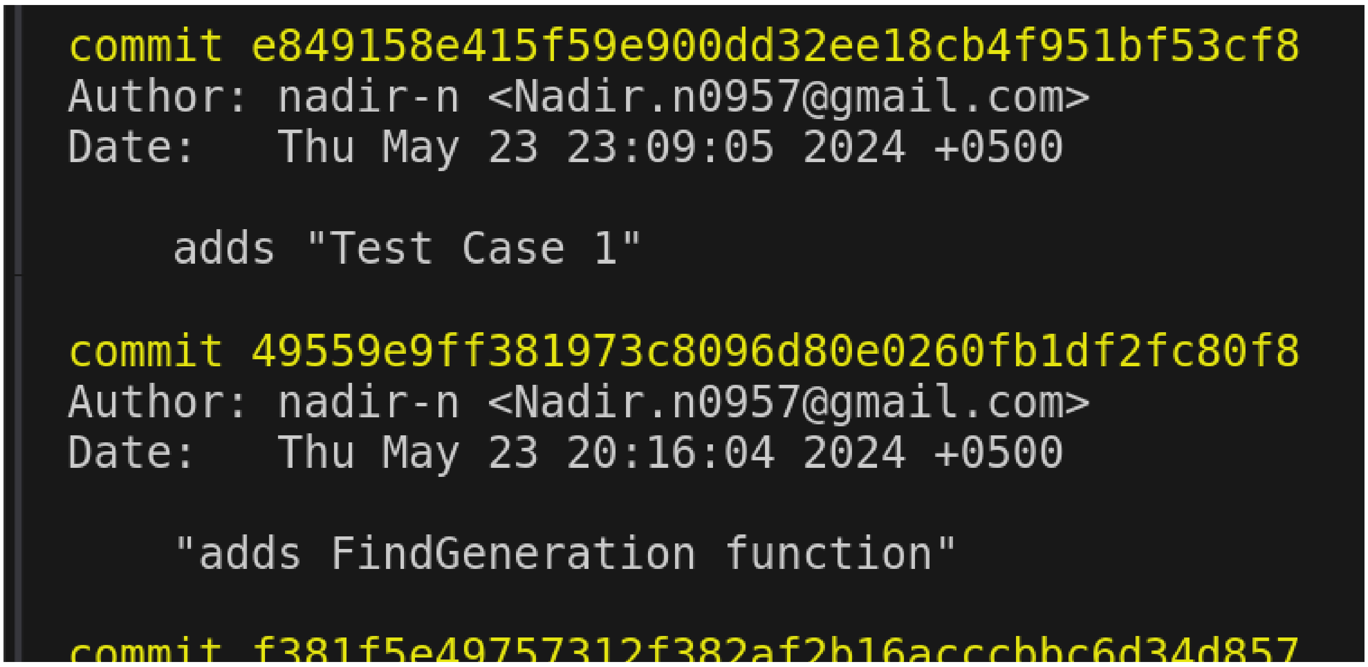
if 2 is pressed then the program asks user to enter date, month and year which is taken as **input** and the LuckyNumber and all other details are then stored in a file named output.txt. In this way, **output is stored in a file**.

The test cases **return** values and take as **parameters**. So All requirement are fulfilled.

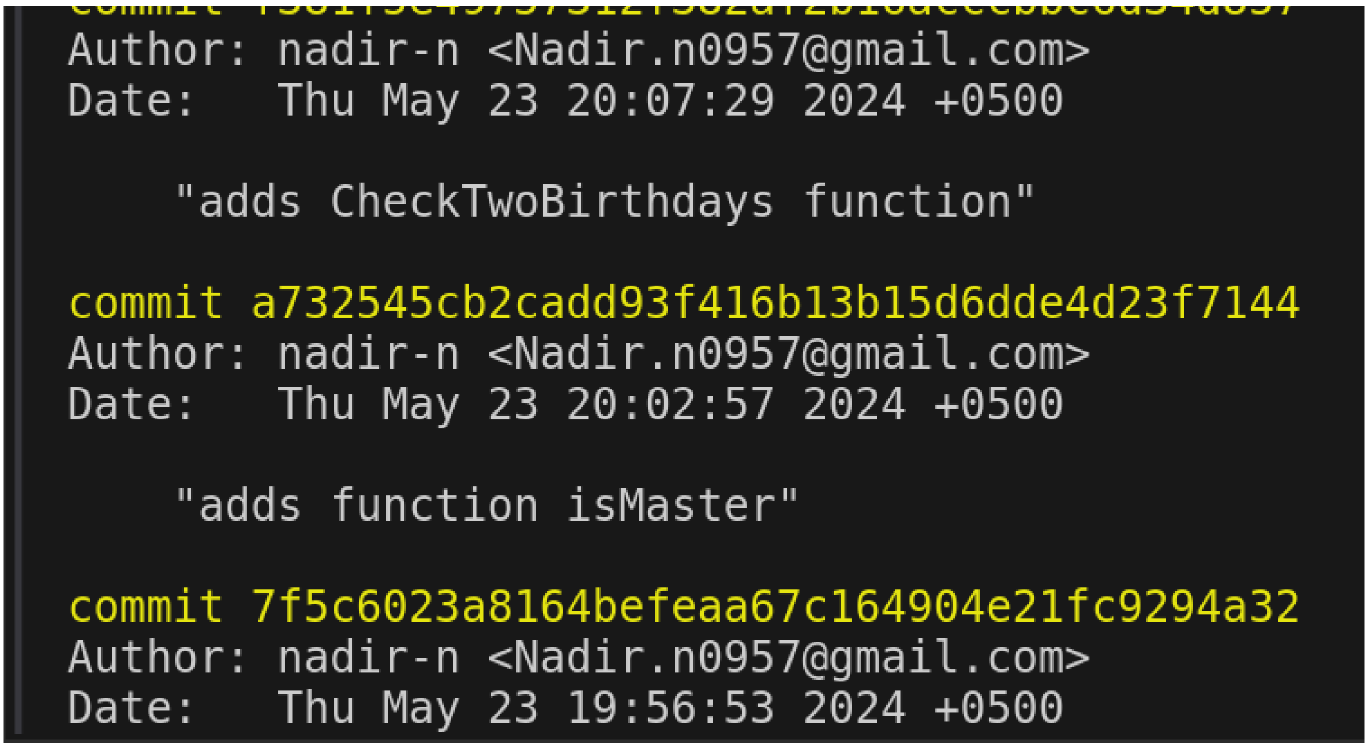
Version Control



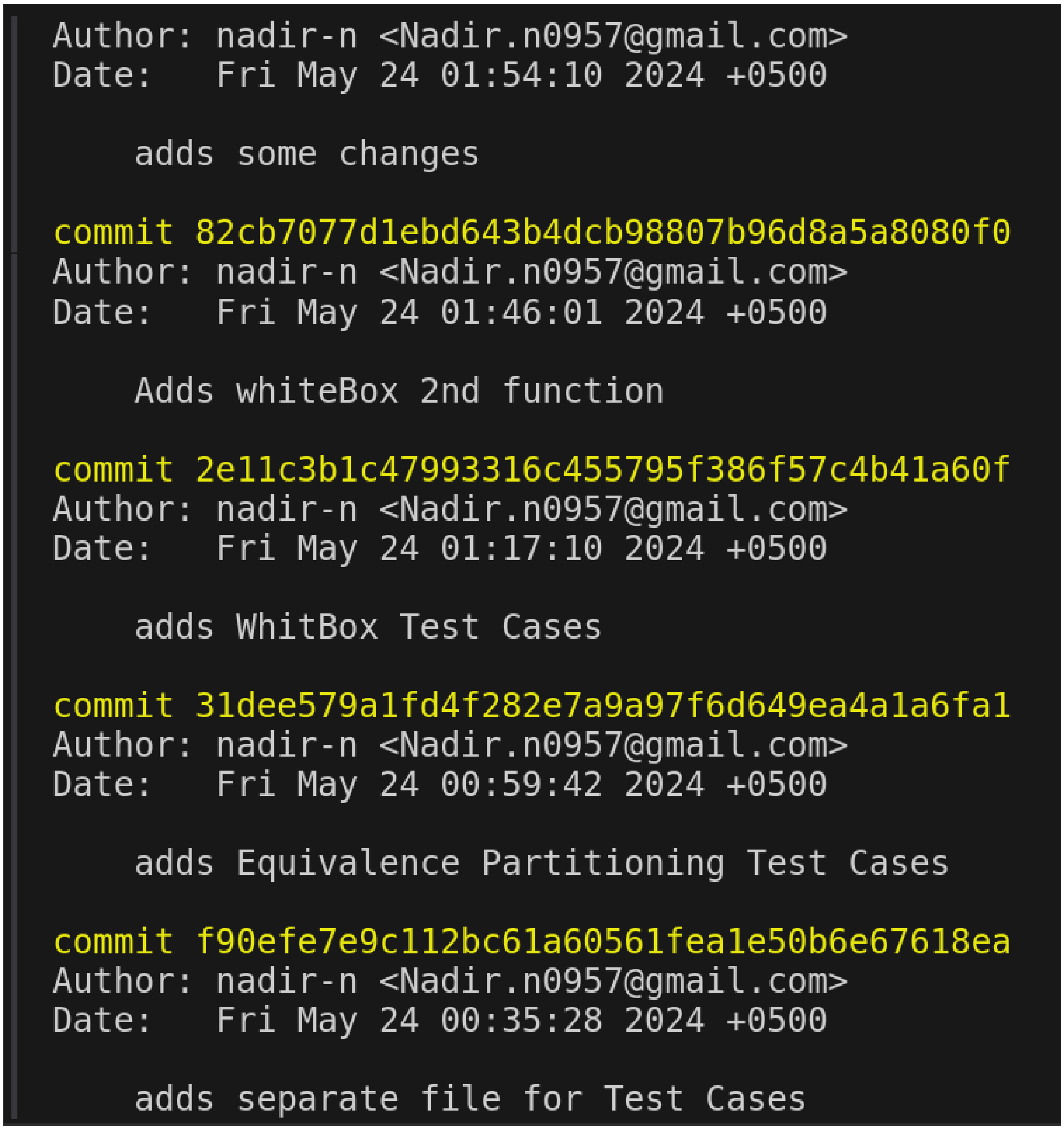
```
Date:   Thu May 23 19:56:53 2024 +0500

    "adds lucky colour function"

commit 90d8a193cd8a7836a3ae88ef378913e6e3cc1a55
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Thu May 23 19:50:07 2024 +0500

    "adds master number condition"

commit a6ea3d6a1e4037f3e4d2e0493adf63f0bf7acf8d
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Thu May 23 19:44:16 2024 +0500

    "adds LifePathNumber with functions"

commit 8b7f8964aab851f06fba3f1f4b1d9645d272d128
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Thu May 23 19:16:52 2024 +0500

    "adds LifePathNumber functionality"

commit 55359401f1dc3ca375278ec8712874bc67fe25d0
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Thu May 23 18:21:06 2024 +0500

    "testing repo"
```



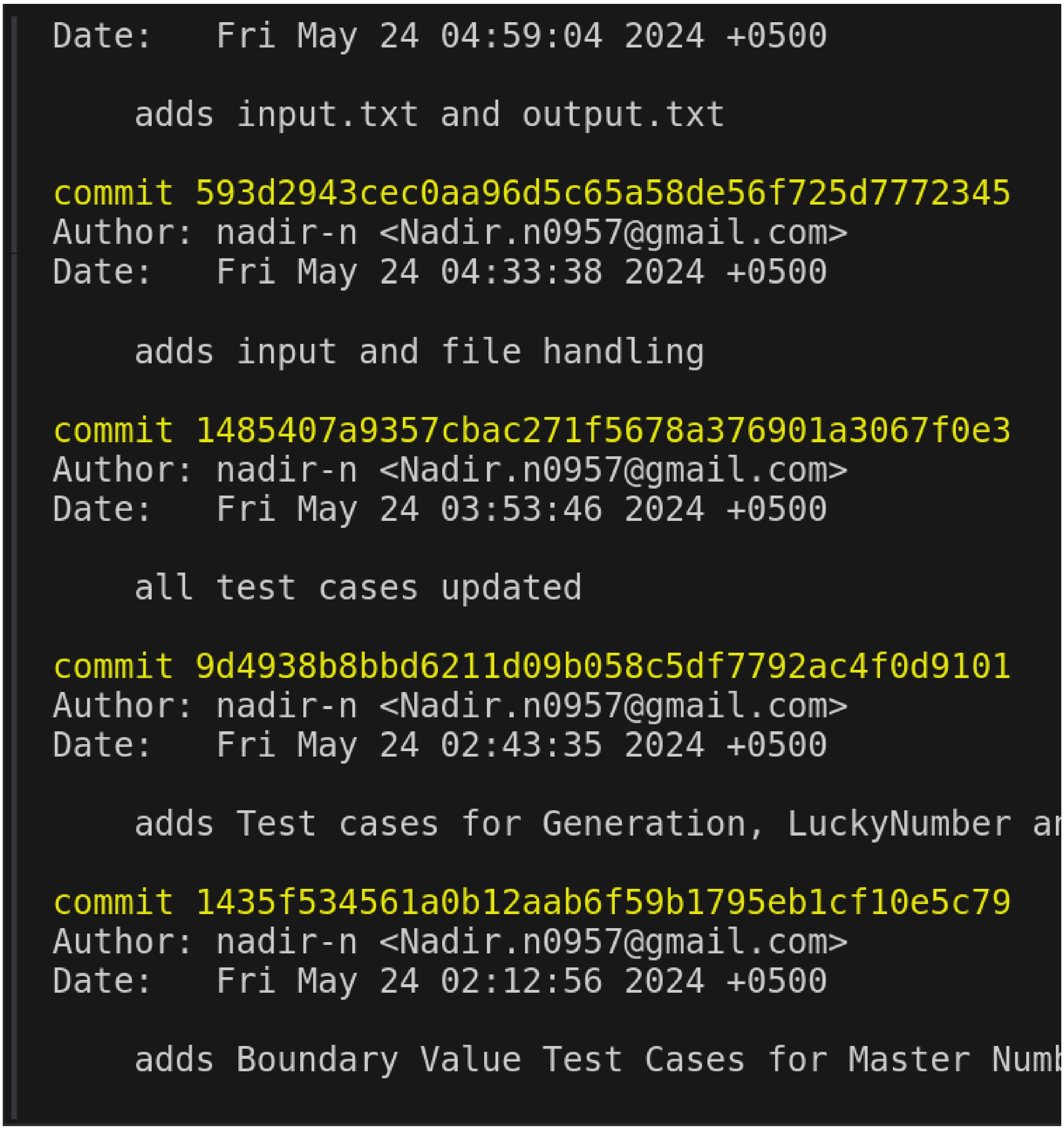
```
commit e849158e415f59e900dd32ee18cb4f951bf53cf8
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Thu May 23 23:09:05 2024 +0500

    adds "Test Case 1"

commit 49559e9ff381973c8096d80e0260fb1df2fc80f8
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Thu May 23 20:16:04 2024 +0500

    "adds FindGeneration function"

commit f381f5e49757312f382af2b16acccbbc6d34d857
```



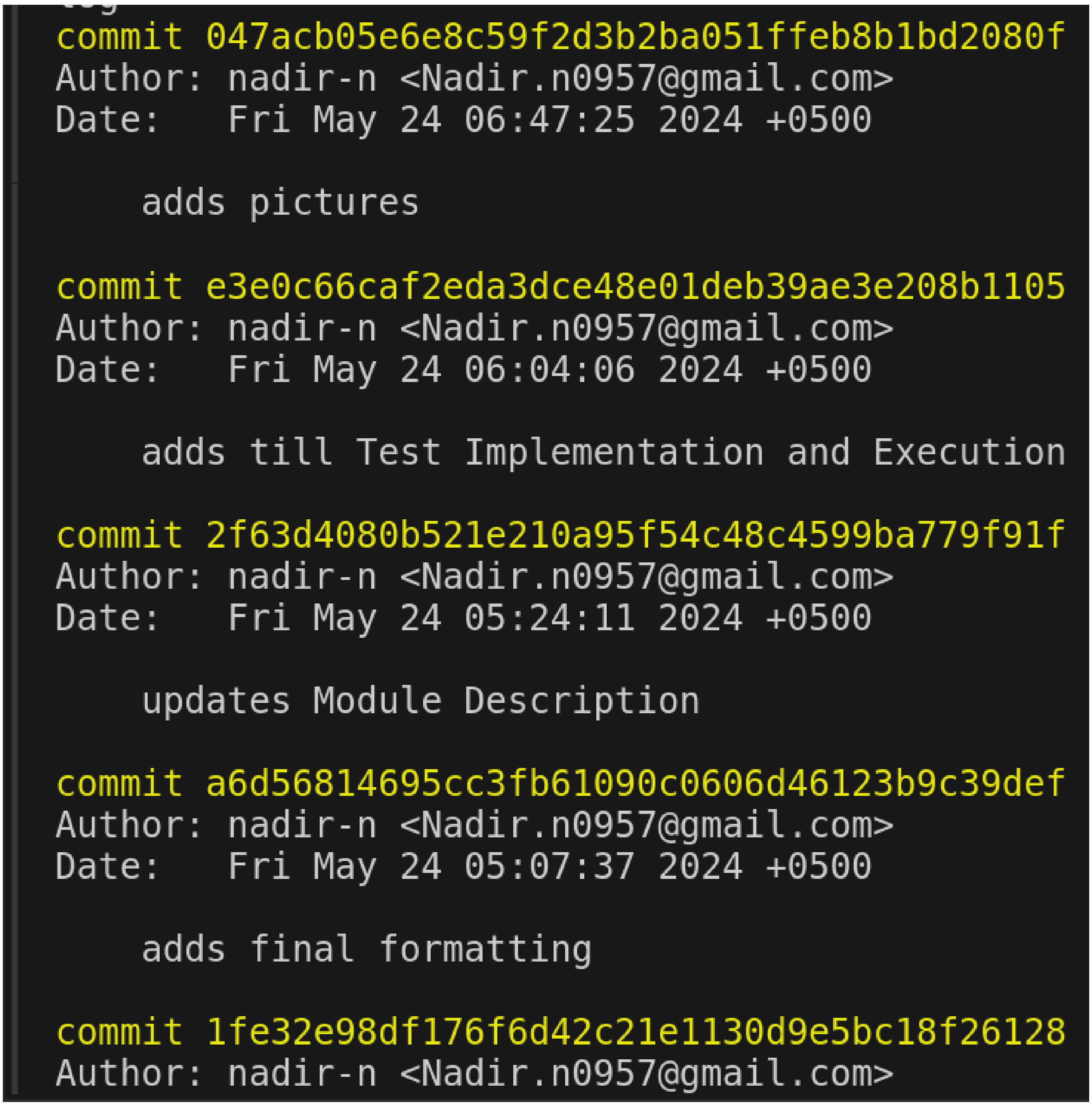
```
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Thu May 23 20:07:29 2024 +0500

    "adds CheckTwoBirthdays function"

commit a732545cb2cadd93f416b13b15d6dde4d23f7144
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Thu May 23 20:02:57 2024 +0500

    "adds function isMaster"

commit 7f5c6023a8164befeaa67c164904e21fc9294a32
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Thu May 23 19:56:53 2024 +0500
```



```
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Fri May 24 01:54:10 2024 +0500

    adds some changes

commit 82cb7077d1ebd643b4dcb98807b96d8a5a8080f0
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Fri May 24 01:46:01 2024 +0500

    Adds whiteBox 2nd function

commit 2e11c3b1c47993316c455795f386f57c4b41a60f
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Fri May 24 01:17:10 2024 +0500

    adds WhitBox Test Cases

commit 31dee579a1fd4f282e7a9a97f6d649ea4a1a6fa1
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Fri May 24 00:59:42 2024 +0500

    adds Equivalence Partitioning Test Cases

commit f90efe7e9c112bc61a60561fea1e50b6e67618ea
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Fri May 24 00:35:28 2024 +0500

    adds separate file for Test Cases
```



```
Date:   Fri May 24 04:59:04 2024 +0500

    adds input.txt and output.txt

commit 593d2943cec0aa96d5c65a58de56f725d7772345
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Fri May 24 04:33:38 2024 +0500

    adds input and file handling

commit 1485407a9357cbac271f5678a376901a3067f0e3
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Fri May 24 03:53:46 2024 +0500

    all test cases updated

commit 9d4938b8bbd6211d09b058c5df7792ac4f0d9101
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Fri May 24 02:43:35 2024 +0500

    adds Test cases for Generation, LuckyNumber ar

commit 1435f534561a0b12aab6f59b1795eb1cf10e5c79
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Fri May 24 02:12:56 2024 +0500

    adds Boundary Value Test Cases for Master Numb
```



```
commit 047acb05e6e8c59f2d3b2ba051ffeb8b1bd2080f
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Fri May 24 06:47:25 2024 +0500

    adds pictures

commit e3e0c66caf2eda3dce48e01deb39ae3e208b1105
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Fri May 24 06:04:06 2024 +0500

    adds till Test Implementation and Execution

commit 2f63d4080b521e210a95f54c48c4599ba779f91f
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Fri May 24 05:24:11 2024 +0500

    updates Module Description

commit a6d56814695cc3fb61090c0606d46123b9c39def
Author: nadir-n <Nadir.n0957@gmail.com>
Date:   Fri May 24 05:07:37 2024 +0500

    adds final formatting

commit 1fe32e98df176f6d42c21e1130d9e5bc18f26128
Author: nadir-n <Nadir.n0957@gmail.com>
```

Discussion

The comprehensive black-box testing approach, encompassing equivalence partitioning and boundary value analysis, ensured robust validation of the software's functionality across a wide range of input scenarios. Notably, the tests revealed accurate handling of edge cases such as invalid dates and boundary conditions.

White-box testing provided valuable insights into the internal logic and decision-making processes of the software, particularly in modules such as CheckTwoBirthdays and LifePathNumber. By examining the code paths and conditional statements, potential errors and discrepancies were identified and effectively addressed.

Overall, the test results demonstrate the software's resilience and accuracy in calculating Life Path Numbers, determining lucky colors, identifying master numbers, and assigning generations based on birthdates. The meticulous test design and execution process underscore the software's reliability and suitability for numerology analysis tasks.
