1. brief description of notable obstacles you overcame. (In Project 1, for example, some people had the problem of figuring out how to work with more than one version of a program in Visual C++.)

I had to initialize some variables and use getline for strings. After running my code in the g31++ compiler I learned that I should put braces after each if, if-else statement in my code. Initially, I had unused Boolean variables. But g31++ said that I should erase them since they were not used for anything and that is what I did.

When starting the project, I intended to make an enum class to assign months and to write a switch statement to determine the month code. However, I could not use a cin statement to make the user input a value of type enum. So, I opted to just use if else statements and Boolean expressions that check if the string is a correctly spelled month. Furthermore, I had multiple logic errors because I coded my program in one sitting. However, I was able to fix them through investigating codeboard test cases which my program failed at the time.

1. A list of the test data that could be used to thoroughly test your program, along with the reason for each test. You don't have to include the results of the tests, but you must note which test cases your program does not handle correctly. (This could happen if you didn't have time to write a complete solution, or if you ran out of time while still debugging a supposedly complete solution.)

Note: Because the question says the results if the tests do not have to be included, month and century code test cases only had an input and no output.

Special Test Case (Invalid Dates): My program does not give an error for dates such as February 30th 2020, April 31st 2020, etc… thus, these days are classified as valid days and give a valid weekday according to the program. My program assumes that the user will avoid inputting dates that do not exist in real life since it is of no use.

Test Case 1 (Wrong Month Input):

Provide a month: **january**

Provide a day: **2**

Provide a year: **2020**

Invalid month!

Test Case 2 (Wrong Month Input):

Provide a month: **January january**

Provide a day: **2**

Provide a year: **2020**

Invalid month!

Test Case 3 (Wrong day input (Day < 1)):

Provide a month: **February**

Provide a day: **-10**

Provide a year: **2000**

Invalid day!

Test Case 4 (Wrong day input (Day > 31)):

Provide a month: **February**

Provide a day: **32**

Provide a year: **2000**

Invalid day!

Test Case 5 (Wrong Year Input (Year < 1)):

Provide a month: **March**

Provide a day: **3**

Provide a year: **-101**

Invalid year!

Test Case 6 (Wrong Year Input (Year > 3000)):

Provide a month: **January**

Provide a day: **29**

Provide a year: **3001**

Invalid year!

Test Case 7 (Order of Invalid input message):

Provide a month: **-101**

Provide a day: **-101**

Provide a year: **-101**

Invalid month!

Test Case 8 (Order of Invalid input message):

Provide a month: **March**

Provide a day: **-101**

Provide a year: **-101**

Invalid day!

Test Case 9 (Order of Invalid input message):

Provide a month: **March**

Provide a day: **29**

Provide a year: **-101**

Invalid year!

Test Case 10 (Leap Year divisible by 4):

Provide a month: January

Provide a day: 5

Provide a year: 8

January 5, 8 was a Saturday!

Test Case 11 (Not a Leap Year divisible by 100):

Provide a month: January

Provide a day: 5

Provide a year: 100

January 5, 100 was a Tuesday!

Test Case 12 (Leap Year divisible by 400):

Provide a month: January

Provide a day: 5

Provide a year: 400

January 5, 400 was a Wednesday!

Test Case 13 (correct month input January):

Provide a month: January

Provide a day: 5

Provide a year: 400

January 5, 400 was a Wednesday!

Test Case 14 (correct month input February):

Provide a month: February

Provide a day: 5

Provide a year: 400

Test Case 15 (correct month input March):

Provide a month: March

Provide a day: 5

Provide a year: 400

Test Case 16 (correct month input April):

Provide a month: April

Provide a day: 5

Provide a year: 400

Test Case 17 (correct month input May):

Provide a month: May

Provide a day: 5

Provide a year: 400

Test Case 18 (correct month input June):

Provide a month: June

Provide a day: 5

Provide a year: 400

Test Case 19(correct month input July):

Provide a month: July

Provide a day: 5

Provide a year: 400

Test Case 20 (correct month input August):

Provide a month: August

Provide a day: 5

Provide a year: 400

Test Case 21 (correct month input September):

Provide a month: September

Provide a day: 5

Provide a year: 400

Test Case 22 (correct month input October):

Provide a month: October

Provide a day: 5

Provide a year: 400

Test Case 23 (correct month input November):

Provide a month: November

Provide a day: 5

Provide a year: 400

Test Case 24 (correct month input December):

Provide a month: December

Provide a day: 5

Provide a year: 400

Test Case 25 (Century Code= -2):

Provide a month: January

Provide a day: 5

Provide a year: 400

Test Case 26 (Century Code= 3):

Provide a month: January

Provide a day: 5

Provide a year: 500

Test Case 27 (Century Code= 1):

Provide a month: January

Provide a day: 5

Provide a year: 600

Test Case 28 (Century Code= -1):

Provide a month: January

Provide a day: 5

Provide a year: 700

Test Case 29 (correct day output Monday):

Provide a month: January

Provide a day: 18

Provide a year: 2021

Test Case 30 (correct day output Monday):

Provide a month: January

Provide a day: 18

Provide a year: 2021

Test Case 31 (correct day output Tuesday):

Provide a month: January

Provide a day: 19

Provide a year: 2021

Test Case 32 (correct day output Wednesday):

Provide a month: January

Provide a day: 20

Provide a year: 2021

Test Case 33 (correct day output Thursday):

Provide a month: January

Provide a day: 21

Provide a year: 2021

Test Case 34 (correct day output Friday):

Provide a month: January

Provide a day: 22

Provide a year: 2021

Test Case 35 (correct day output Saturday):

Provide a month: January

Provide a day: 23

Provide a year: 2021

Test Case 36 (correct day output Sunday):

Provide a month: January

Provide a day: 24

Provide a year: 2021