Mit Chan
UC Innovation Quiz
Question 2 and 3

One of the requirements of a website is to be able to enter partial birth dates, either by entering the day, month, or year, or any combination of them. Once entered, the website needs to display the birthdate in month/day/year format as well as the age. A screenshot is given below.

Please list all the test cases that you would need to test, as well as expected results, to make sure the website is implemented correctly.

Assumptions:

- "Birth Month" input field accepts the month as either a string from {"January", "February", …, "December"} which maps to [1,2,...,12] respectively OR a number as a string in the range [1,2,...12].
- The user must enter at least one input for the day/month/year. They can enter any combination or partial combination, but cannot enter empty inputs for all of them.
- If the user does not enter a year, their age will not be displayed.
- If the user does not enter a month,
  their age will be calculated only by the current year - inputted year.
- The birthdate format follows:
  - MM/DD/YYYY (All inputs provided)
  - If the user does not input a certain field, then that field will be shown with its initials.
    - e.g) Input: Month: July, Day: 4, Year: null == Birthdate: 7/4/YYYY

Constraints:

- Birth Month: {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"} OR {'1','2','3','4','5','6','7','8','9','"0","11","12"}
- 1 <= Day in Month <= 28,29,30,31 (For different months)
- 1900 <= Birth Year <= Current Year (2021)

Test Cases:

Partial Combinations

Description	Example	Expected Output
Month and Year Combination	Month: 12 Day: null Year: 2017	Birthdate: 12/DD/2017 Age: 3

Day and Year Combination	Month: null Day: 15 Year: 2017	Birthdate: MM/15/2017 Age: 4
Month and Day Combination	Month: July Day: 10 Year: null	Birthdate: 7/10/YYYY Age: null
Year only	Month: null Day: null Year: 2010	Birthdate: MM/DD/2010 Age: 11
Day only	Month: null Day: 25 Year: null	Birthdate: MM/25/YYYY Age: null
Month only	Month: December Day: null Year: null	Birthdate: 12/DD/YYYY Age: null

Day Test Cases

Description	Example	Expected Output
Day does not exist in month (Repeated for each month with its respective last day) e.g) Last day of March is 31 Last day of April is 30 and use these last days as the upper bound for days.	Month: February Day: 29 Year: 2017	Error - February does not have a 29th day in 2017.
Leap Year is ok	Month: February Day: 29 Year: 2020	Birthdate: 2/29/2020 Age: 1
Invalid Day (Negative Number)	Month: null Day: -1 Year: 2018	Error - Given day not in range(1, 31)
Zero'th day (Regarded as empty)	Month: 12 Day: 0 Year: 2018	Birthdate: 12/DD/2018
Invalid Day (String)	Month: null Day: Abc Year: 2020	Error - Given day is not a number.

Month Test Cases

Description	Example	Expected Output
Invalid Month (Numerical)	Month: 13 Day: 29 Year: 1998	Error - Month not in range (1,12).
Invalid Month (String) -	Month: Dacadeber Day: null Year: null	Error - Month not recognized as a valid month.
Valid Month (Numerical)	Check every number from 1 through 12 in the month input	Birthdate: 1-12 in the month field
Valid Month (String)	Check every string from the set {“January”, “February”, … “December” } (lowercase is ok)	Birthdate: Respective month mapped to its numerical number.

Year Test Cases

Description	Example	Expected Output
Invalid Year (Greater than current year)	Month: February Day: 2 Year: 2201	Error - Inputted year is greater than the current year.
Less than minimum year - Invalid Year	Month: null Day: null Year: 1800	Error - Inputted year is less than the current year
Valid Year (In range from 1900-2021)	Month: December Day: 14 Year: 1984	Birthdate: 12/14/1984 Age: 36

Assume that a program has over 1 billion lines of code. During production, the program crashes. Without using a debugger, describe a possible debugging process or approach that could be used to pinpoint the exact line that is causing the segmentation fault.

Because the program is crashing due a segmentation fault, I would look to specific areas in the code where memory is constantly being written to, referenced, dereferenced, or accessed. Since there are over 1 billion lines of code, this may not be very helpful, but assuming I had a working knowledge of the codebase, I could pinpoint certain areas where memory problems could occur.

Some other considerations I would identify before continuing:

- Does our program use multithreading?
  - Yes: Make sure to monitor thread id's and observe if locking behavior is as expected.
- Does our program read/write from any connected database?
  - Yes: Make sure read-only memory is not being written to.
- Do we have version control?
  - Yes: Look to previous versions first, comparing and identifying sections of code that have been added, changed, or removed. I'm not sure if this would be against the condition of using a debugger since to my knowledge, version control software like Git also has debugging tools to find introduced bugs in commits.
- Do we have system calls implemented?
  - No: Implement system call monitoring to see system calls by the program.

After identifying areas of interest, I would recompile the program and add the following: additional logging by writing to a log file to record events happening in the program, unit tests to test basic expected functionality and structure of program classes, structures, etc, if they were not already written, and assert statements to verify program behavior. In writing to log files, I would include the time, thread-ids if our program uses multithreading, and structures being used to identify out of the ordinary values throughout the program's runtime. Doing so will help pinpoint the exact line that is causing the segmentation fault by revealing unexpected behavior.

Finally, although there are 1 billion lines of code, I would sit down with the source code and analyze it, running through the program - or at least areas of interest - assuming it is feasible to understand the flow and behavior of the program.