A. Titles with MPL Quizzes i.e. Code Question Brix

1. **Pearson** to create MPL course and provide Author directions on accessing the course so the author can pick MPL problems
2. **Author** to pick MPL problems and arrange it in a spreadsheet for each chapter and section
3. **Pearson** to check if the problems exist on Nemo
    a. Most Java, C++, Python problems are on Nemo
    b. C problems are not on Nemo
    c. If the title is a first time title (like the C language), then Pearson to coordinate with Alicia Anderson to have a Pearson developer write a script to import all the C problems from MPL to Nemo
    d. If a few problems don't exist on Nemo, then Pearson to inform this to Vendor and Vendor to do Step 6b.
4. **Pearson** to create Table of Contents with exact question number in each chapter and section. Pearson to finalize the spreadsheet with exact MPL exercise numbers.
5. **Pearson** to provide Vendor with with the spreadsheet of problems. At least 90% of problems should be present on Nemo otherwise Step 3c.
6. **Vendor** to create Titles Structure on Nemo and MPL quizzes on Nemo based on spreadsheet provided to them in Step 4
    a. 90% of problems should exist on Nemo
    b. For the 10% that may not exist on Nemo, create the problems on Nemo. Refer document that shows how to create a Code Question problem on Nemo.
7. **Pearson** to spot check vendor's work, especially if the vendor created any Code Question problems (i.e. 6b)

B. Titles with author created programming projects (i.e. Liang titles)

1. **Author** to create programming projects in MPL. Pearson to coordinate with author to ensure the questions are being created in the right MPL course.
2. **Pearson** to provide this information (author's MPL username and password and where exactly to find the programming projects) to the vendor
3. **Pearson** to inform vendor when it is time to build the programming projects and where exactly it should be placed on Nemo.
4. **Vendor** to create the programming projects on Nemo and associate it with the title under section "programming projects". Refer document that shows how to create a Code Question problem on Nemo. Programming Projects are nothing but Code Questions that are placed at the end of every chapter.

5. **Pearson** to monitor step 4 and make sure the vendor is placing it in the right title and right sections.
6. **Pearson** to spot check vendor's work, including font, copy editing, spacing.

C. Titles with Simple Code Questions (i.e. Gaddis)

1. **Pearson** to go through the manuscript and check if the Simple Code Questions seem buildable
    a. Should the question be a simple code?
    b. Is the answer small enough to fit in the simple code editor displayed on Revel?
    c. Modify the question if necessary so that the student answers only one line of code. (Ideally the person should be able to make this decision on their own. If not, they can check with the creator of the manuscript.)
2. **Pearson** to make a list of all the simple code questions and provide the vendor
3. **Vendor** to create the questions on Turings' Craft.
4. **Vendor** and **Pearson** to work very closely at this stage:
    a. There could be questions that cannot be coded on Turings' Craft for technical reasons. Pearson would need to make decisions about that such as change the question or modify it or see if another programming language can be used behind the scenes
    b. There could be questions for David Arnow that Pearson will need to coordinate getting answers
5. **Pearson** to spot check the questions on Tunrings' Craft platform.
6. **Pearson** to inform vendor when it is ready for creation as Brix on Nemo and give vendor instructions on creating the Brix
7. **Vendor** to create the Brix on Nemo. Refer document that shows how to create a Simple Code Question problem on Nemo.
8. **Pearson** to monitor step 7 and make sure the vendor is placing it in the right title and right sections and using the right exercise numbers.
9. **Pearson** to spot check vendor's work, including font, copy editing, spacing.

D. Titles with Live Examples (i.e. Gaddis)

1. **Pearson** to go through the manuscript and check if the Live Examples seem buildable
    a. If it's Python, it cannot be interactive. It should be changed to full programs even though that is not how Python is used in the real world.
    b. File operations are not supported (can write into a file, cannot read)

    c. Java programs should all be in one class with one main method
2. **Pearson** to finalize list of Live Examples and provide this to vendor
3. **Vendor** to build the Live Example on Nemo
4. **Pearson** to monitor step 3 and make sure the vendor is placing it in the right title and right sections and using the right names for titles.
5. **Pearson** to spot check vendor's work, including font, copy editing, spacing. If the answer instruction states "change the comment on line 5" then Pearson to check that there is indeed a comment on line 5 i.e. copy-pasting has not resulted in different line numbers due to spaces in code.

E. Titles with Animations (i.e. Gaddis)

1. **Pearson** to provide vendor with book PDF and a spreadsheet which lists all the exercises that should be animated. Sometimes the author has the source code, in which case Pearson can provide this source code to the vendor so that the vendor doesn't need to type out the whole program.
2. **Vendor** to create animations and share it with Pearson. If there are any questions about animations, Vendor and Pearson to be in touch. Often Pearson will need to clarify with the author.
3. **Pearson** to coordinate with GEX to get it uploaded on media server. Also spot check a few animations.

F. What to do when there is a Code Question (i.e. MPL) Brix content bug

1. **Pearson** to check if the bug can be reproduced on Revel and MPL
    a. If the bug can be reproduced on MPL, then contact David Arnow
    b. If not, step 2
2. **Pearson** to check if the exercise numbers on Author Page on Nemo matches the exercise number on MPL. This is usually the cause for the problem. If it doesn't match, Pearson to fix it and publish the problem.
3. If David Arnow makes any change to the problem on MPL, then **Pearson** to make the same changes on Nemo and publish.

G. What to do when there is a Simple Code Question Brix content bug

1. **Pearson** to check if the bug can be reproduced on Revel and MPL
    a. If the bug can be reproduced on MPL, then contact **Vendor**
    b. If not, step 2
2. **Pearson** to check if the exercise numbers on Author Page on Nemo matches the exercise number on MPL. This is usually the cause for the problem. If it doesn't match, Pearson to fix it and publish the problem.

3. If **Vendor** makes any change to the problem on MPL, then **Pearson** to make the same changes on Nemo and publish.
4. If the bug is a more of a suggestion from user than a bug, then **Pearson** to decide whether it should be fixed. Vendor does not make this decision.

H. What to do when there is a Live Code Example Brix content bug

1. **Pearson** to check if the Brix matches the manuscript. If not, fix it.
2. If the Brix matches the manuscript, and the customer points out an error, then **Pearson** to fix it. (Ideally the person should be able to make this decision on their own. If not, they can check with the creator of the manuscript.)
3. If the error states that it isn't compilable, then make sure the right compiler is chosen in the Author Tab. In short, Live Code Example errors are usually straightforward that **Pearson** can fix it without vendor's or author's help.

I. What to do when there is an animation bug

1. **Pearson** to get the file from the media server. If it's a typo or word change, it's easy to change the word across all the files. Pearson can change it themselves or ask the vendor to change it. If it's a major change, Pearson to provide the files to the vendor and **Vendor** to change the animation.