Report for Assignment-2 Part 3

The initial idea and the requirements for this assignment are that we should be able to implement undo operation after a last valid writeat. For this, it was given that a writeat will not immediately write to the file,rather it will store the data entered and will write to the file after another writeat is executed or if the file was closed. For this, initially in the Assignment-2 Part-1, I have implemented delaying of the writeat function using extra variables that store the data entered for that particular writeat instance. Using this, I was able to implement undo by clearing the stored variable values when an undo is executed, by this the data that is provided to the writeat is cleared off an this gets into the initial state where a writeat is not executed.

As part of Assignment-2 Part-3, after looking at the reference monitors and the attack cases of others, I identified that I have missed to implement another requirement. That is apart from the ability to be undone and potentially delayed writeats, all writeat operations should behave the same way as they do in the RepyV2 API. So, for this I went through the attack cases that were able to go through my reference monitor and I had found the list of cases which my reference monitor was not defending against. I have listed out these from my identification, my reference monitor was not able to handle negative offset for writeat, was not able to handle seek past end of file for writeat, was not handling invalid data being provided to writeat, was not correctly handling executing an undo after fresh new file is opened and before a writeat is being executed. Apart from these, the readat would be the same, and we would be required to add a writeat in the close function as the data is written into the file after a file is closed.

For these issues, I have fixed them as follows. For the negative offset and invalid offset being provided, I have implemented an if block checking if the offset provided is <0 or if it is None or if it is an empty string, then raising a RepyArgumentError if it falls under those cases. If the data provided to the writeat is also invalid, that is of the type None, then I have implemented an if block for this case also to throw a RepyArgumentError if it falls under this case. For the case where a writeat is being executed for an already closed file, I have implemented a flag that stores the value for if the particular file is open or closed, through this I was able to check if the file was already closed and was able to throw a FileClosedError. If the offset provided is greater than the size of file, then I have created a variable which stores the size of file and compares the offset to that and then throws a SeekPastEndOfFile error if it falls under that case. During testing my modified reference monitor against all the test cases, I again found one error which is the order of precedence of the exceptions in repyV2, so accordingly I have implemented the exception checks in the order that is required for repyV2 as mentioned in its API. And, in the undo I have found that we cannot execute an undo if the file is closed, so I have implemented an if block to check for this case. These changes allowed my reference monitor to be able to defend against almost all attack cases.