My accuracy number is 11 resolved instances out of 20. I have observed that providing the right context (detailed function comments about behavior, returns and arguments, a prompt that suggests a kind of workflow, even maybe function names) makes a huge difference in the accuracy of the agent. I struggled to get the agent to use all the tools that I provided, though. It used several heavily and didn't touch others. I believed that all my tools held merit, so I tried to encourage use of them by renaming the functions, adding information in the function comment about how the tool could be helpful, and adding text into the prompt telling the agent to use all the provided tools. None of this actually worked, though. I still do not understand if this means that the unused tools aren't good or if there is some other way I can entice the agent to use them. These are descriptions of the custom tools I created:

- show_file(self, file_path: str)- Shows the contents of the specified file. I included this because it was suggested in the repo. I observed this tool to be heavily used.

- show_files(self, file_paths: list[str])- Shows the contents of all files in the given list of file paths. I made some tools that return a list of file names, and I thought it might save effort for the agent to be able to print out all the contents at once rather than have to iterate through them all. I observed this tool to be lightly used.

- search_files(self, content: str)- Return a list of files which contain the given content string and the few lines of code including and surrounding each instance (with line numbers). It was suggested for us to read the live-swe-agent paper. I implemented this tool as a result of reading that paper. I experimented a bit with how much context to provide around each instance and found that including the 5 lines of code before and after each occurrence yielded the best results. I included all line numbers so that the agent would have max context for any future actions. I observed this tool to be heavily used.

- find_text_occurrences_in_file(self, file_path: str, content: str)- Return a list of all line numbers and instances in the given file where the given content appears. Includes context of line before and after the instance. This tool was not really used. I think maybe in the end search_files provides this same info and the agent would do a broad search before narrowing down to a single file anyway (as I added to the prompt). This was inspired by live-swe-agent, though, so I was honestly disappointed that my agent didn't use it.

- extract_import_statements_from_file(self, file_path: str)- Returns all the import statements in the given file. Again, I got this from live-she-agent but my agent didn't use it. It seemed like a promising way for the agent to gain context about the codebase and possible causes of bugs, though.

- list_all_python_files(self)- Returns a list of all python files via a recursive search though the repo. Excludes hidden files. This was heavily used by my agent! I see it as a great starting point for learning which files are worth checking out.

- list_python_files_with_todos(self)- This was a long shot. I didn't read the test materials, but if they happened to contain unfinished code this would have been an awesome tool. My agent didn't touch it, though.

- list_test_python_files(self)- I don't know why my agent didn't use this! It seemed useful to provide a way for it to run tests to get more context about where failures exist.

I had a couple more functions that I wrote but took out of the environment because I felt they hurt execution. Functions to insert content, delete content and replace content in files. My agent used the replacement mostly, but I spent a long time fine tuning and even though specific test instances looked great to me, performance actually suffered. There was also a function to run a python script that might have been used once or twice but didn't seem to affect performance. I also tried out a function to count all the functions in a python file to provide information about the busiest files. I think it makes sense that bugs would be in busy files.

Overall this was a lot of trial and error. Things that I thought would help (mostly tools) didn't, and things that seemed unnecessary to me (prompt changes, comment changes) made a world of difference. My takeaway is that even though the LLM talks like me, it does not think like me. If I want to be able to use LLMs and agents effectively, I am going to need to learn more about how to get the responses and behaviors that I desire.