Gnarly Bugs Submission Format

Criteria for submission:

- Contains a bug that would take at least 6 hours for an experienced programmer to solve, and ideally >20hrs
    - More specifically, ">6 hours for a decent engineer who doesn't have context on this particular codebase". E.g. a randomly selected engineer who's paid $100-$200 per hour who's familiar with the language and overall stack that's being used, but not the person who wrote the code, and not an expert in the particular component that is causing the bug.

- Ideally, has not been posted publicly in the past
    - (Though note that we may still accept submissions from public repositories given that they are not already in a SWE-bench dataset and meet the rest of our requirements. Check with us first.)
- You have the legal right to share it with us (e.g. please don't send us other people's proprietary code or anything you signed an NDA about)
- Ideally, the task should work well with static resources - e.g. you can have a local copy of the documentation for all the relevant libraries, or some other information, but don't have general internet access.
    - This is because we want to make sure the difficulty doesn't change over time, if e.g. someone posts a solution to stack overflow or whatever.

- Ideally, the codebase is written in Python but we will accept submissions written in other languages.

More context and guidance:

- the eval will involve the model actively experimenting and trying to debug; it doesn't have to be something where you can solve it by just reading the code

- complexity is generally good (e.g. multiple files + modules, lots of interacting parts), but ideally it should be easy to run the task without needing to spin up a lot of resources. Installing packages or starting a local server is fine, using a GPU is somewhat annoying.

- All of these are valid types of task:
    - The goal of the task isn't to diagnose + directly solve the bug, it's just to get the code working; sidestepping the bug is a valid solution
    - You need to identify the specific line of code that is wrong and explain why the problem is happening
    - There are multiple bugs that need to be solved

To make it easy for us to convert your gnarly bug into a task that we can use in our evaluations, we require that the submission be in a particular format and contain certain documentation.

When you are done:

- Please send submissions to gnarly-bugs@evals.alignment.org in the form of a zip file.
- Your email should include the number of hours it took for you to get the code from its original state into our required format.

You are welcome to email gnarly-bugs@evals.alignment.org with any questions, including if you are unsure whether your potential submission would meet the criteria.

Format Specification

Your output should be a non password protected zip file with the following file structure:

```
gnarly_bugs_<short descriptive name>_<date of them sending as YYYYMMDD>/
|
├── SWE-bench.json
├── Codebase.zip <if not hosting repo on github, codebase zip right before bugfix>
|
├── Docs/
|   ├── BugWalkthrough.md
|   ├── SetupInstructions.md <if no fork of SWE-bench, bit less preferred>
|   └── TestInstruction.md <if no fork of SWE-bench, bit less preferred>
|   └── ManualTestingInstructions.md  <if no automated regression tests, less preferred>
|
└── AdditionalInfo.json
```

You should provide five things:

1 - SWE-bench.json, a JSON object in the SWE-bench format, with the following keys:

None

```
instance_id: (str) - A formatted instance identifier, usually as
   repo_owner__repo_name-PR-number.
patch: (str) - The gold patch, the patch generated by the PR (minus test-related code), that
   resolved the issue.
repo: (str) - The repository owner/name identifier from GitHub.
base_commit: (str) - The commit hash of the repository representing the HEAD of the repository
   before the solution PR is applied.
hints_text: (str) - Comments made on the issue prior to the creation of the solution PR's first
   commit creation date.
created_at: (str) - The creation date of the pull request.
test_patch: (str) - A test-file patch that was contributed by the solution PR.
problem_statement: (str) - The issue title and body.
version: (str) - Installation version to use for running evaluation.
```

```
environment_setup_commit: (str) - commit hash to use for environment setup and installation.
FAIL_TO_PASS: (str) - A json list of strings that represent the set of tests resolved by the PR
    and tied to the issue resolution.
PASS_TO_PASS: (str) - A json list of strings that represent tests that should pass before and
    after the PR application.
```

(If your submission doesn't include automated tests, you can instead provide a Markdown file describing how to reproduce the bug and check that it's fixed)

If you are providing a copy of the codebase in your submission then you should still provide the following: patch, hints_text, created_at, test_patch (if applicable), problem_statement, version, FAIL_TO_PASS (if providing automated tests), PASS_TO_PASS (if providing automated tests).

2 - AdditionalInfo.json, a json file containing the following information (this is so we can get an idea of how difficult the task is, and is also where you should indicate whether your repo is public or not).

```
{
  "hours_to_debug": <Estimated number of hours to originally fix the bug>,
  "years_experience_at_time_of_debugging": <Years of working experience doing coding / SWE at the time of debugging>,
  "hourly_rate_in_dollars_at_time_of_debugging": <self explanatory>,
  "repo_was_always_private": <Whether the repo has always been private, true or false>
}
```

***3 - BugWalkthrough.md**, a markdown document that explains:*

- *Any relevant background details*
- *The bug itself*
- *The process that you / someone else took in order to track down the bug.*
- *How the bug is fixed*

(This is so we can get people up to speed with the task quickly, which is extremely helpful for scoring agent performance at scale, or quickly making many variants of the task at different difficulty levels.)

4 - Access to the repo in some form. *Choose one of the following (either is fine):*

Either:	If the repo is hosted on GitHub, give the following users at least read access to the repo: - MeganKW - goodrichb
Or:	Provide a zip of the codebase in your submission

5 - A way to test whether the bug has been fixed by any given change. *This should ideally be in the form of automated tests.*

ONE of the following:	
Ideal	Automated tests and a GitHub fork of the SWE-bench repo for easy testing setup. A GitHub fork of the SWE-bench repo with modifications to the files in https://github.com/princeton-nlp/SWE-bench/tree/main/harness to allow running the script https://github.com/princeton-nlp/SWE-bench/blob/main/harness/engine_evaluation.py on the bug you're submitting
A bit less preferred	Automated tests and two markdown files: 1. SetupInstructions.md, which explains how to set up the codebase for development 2. TestInstructions.md, which explains how to run automated tests for the codebase
Less preferred	If your submission does not include automated tests, then provide a markdown file called ManualTestingInstructions.md. This file should contain instructions which allow us to manually check whether the bug has been fixed.