Week 7 - README

Automated Correctness Checking Pipeline

This program takes a Dafny program, finds its loops, synthesizes loop invariants, inserts those invariants into the source code, and runs Dafny on the modified program. The pipeline integrates the the Week 5 parser, Week 6 synthesizer, and Dafny.

Here's the pipeline

```
def build_and_verify(
    """
    Full Week 7 pipeline for a single program:
      1. Parse with Week 5.
      2. Get invariants (JSON or Week 6).
      3. Locate loop (summary-based, then fallback).
      4. Insert invariants (if any).
      5. Optionally run Dafny.
      6. Return a JSON-able summary.
    """
```

Implementation

1. **Parsing the dafny file**: The loop is located using the Week 5 parser. This gives the method name, loop conditions, and method summaries.
2. **Gathering the invariants:** This step decides what invariants we will insert into the program. The invariants are provided in either a JSON file or they are synthesized using the Week 6 invariant synthesizer.
3. **Locating the Loop:** This step tells us where in the code we have to insert the invariants. The program searches for a `while (...)` header, or `for i := start to end` header.
4. **Insertion into Dafny**: Invariants are inserted after the loop header, before the opening brace:

```
def insert_invariants_into_loop(src, loop, invariants):
    # Inserts at loop.header_end_idx:
    # "\n  invariant <inv1>\n  invariant <inv2>"
    inv_lines = "".join(f"\n{loop.indent}  invariant {inv}" for inv in invariants)
    return src[:insert_pos] + inv_lines + src[insert_pos:]
```

5. **Dafny verification:** If the user passes `--run-dafny`, it runs:

```
dafny /compile:0 instrumented_file.dfy
```

and captures:

- whether the verifier succeeded (`0 errors`)
- the raw output

6. **Outputs a JSON summary:** This summary includes - method name, loop condition that was matched, which invariants were inserted, where the new file was written, and verification result.

Example

Input:

```
method Counter(n: int) returns (i: int)
{
  i := 0;
  while (i < n)
    invariant -i <= 0
  {
    i := i + 1;
  }
}
```

Command:

`python pipeline.py Counter.dfy --out Counter_fixed.dfy --run-dafny`

Terminal Output after verification:

```
{
  "method_name": "Counter",
  "loop_condition": "i < n",
  "inserted_invariants": ["-i <= 0"],
  "output_path": "Counter_fixed.dfy",
  "verification": {
    "ok": "true",
    "raw_output": "Dafny program verifier finished with 1 verified, 0 errors"
  }
}
```


The pipeline successfully:

1. Synthesized the invariant `i <= 0` (i.e., `i >= 0`)
2. Inserted it into the loop
3. Verified correctness with Dafny (1 verified, 0 errors)