**Program Documentation for aver\_per\_sim\_2\_haplos.py**

**Steps the program executes chronologically**

1. Import necessary Python modules including csv, os, collections, time, concurrent.futures, multiprocessing, and sys.

2. Record the program start time.

3. Prevent system sleep on Windows using Windows API calls.

4. Define a function process\_roll\_forward\_and\_average\_streaming that computes time-averaged statistics across simulation runs for a specified target column (e.g., heterozygosity), using roll-forward logic to handle post-fixation values.

5. Define a helper function format\_row that formats averaged results into output-ready strings with proper decimal precision.

6. Define a worker function process\_one\_sim\_nr that reads per-generation data for a single SimNr from the input file.

7. In process\_one\_sim\_nr, parse the input file header to build a column index mapping.

8. Convert each data row to appropriate types (int, float, or str) based on column semantics.

9. Filter rows belonging to the target SimNr, leveraging early exit assuming the input is sorted by SimNr.

10. Validate that all required columns are present; skip processing if any are missing.

11. For each of four metrics (freq\_Aa, freq\_Bb, pan\_heteroz, pan\_homoz), apply the corresponding fixation condition and compute averaged time-series using process\_roll\_forward\_and\_average\_streaming.

12. Format the results for each metric and return them in a dictionary grouped by target column.

13. In the main function, verify that the input file "out\_2\_haplos\_per\_gen.txt" exists.

14. Scan the input file to collect all unique SimNr values while enforcing strict ascending order; abort if order is violated.

15. Sort the SimNr list for consistent processing.

16. Define output filenames and their corresponding headers for four result types.

17. Prepare an output specification mapping each metric to its output file.

18. Initialize all four output files with their respective headers.

19. Create a list of arguments for parallel processing, one per SimNr.

20. Set the multiprocessing start method to 'spawn' for cross-platform compatibility.

21. Launch parallel workers using ProcessPoolExecutor to process each SimNr independently.

22. Collect all results from workers into a dictionary keyed by SimNr and metric.

23. After all workers complete, write results to each output file in strict SimNr order to ensure reproducibility.

24. Print the names of all generated output files.

25. Print the current working directory.

26. Print the total program execution time.