Agentic Benchmark Checklist *v1*

Outcome Validity			
Task Type	**Eval Method**	**Question**	**Score**
Information Acquisition	{Whole String, Substring} Matching	`I.a.1`: Considers expressions semantically equivalent to ground truth.	
		`I.a.2`: Handles redundant words used by agents.	
	Substring Match	`I.b.1`: Handles negation modifiers used by agents.	
		`I.b.2`: Is robust against systematically listing all possible answers.	
		`I.b.3`: Ground truth is sufficiently complex to prevent guessing.	
	LLM-as-a-Judge	`I.c.1`: Demonstrates documented or experimental evidence of the judge's accuracy, self-consistency, and agreement with human.	
		`I.c.2`: Is designed to resist adversarial inputs and reward hacking.	
Code Generation	{Unit, End-to-End} Testing	`I.d.1`: Verifies test cases for correctness and quality (e.g., by human).	
		`I.d.2`: Measures quality of test cases using objective metrics (e.g., code coverage, cyclomatic complexity control).	
	Fuzz Testing	`I.e.1`: Addresses potential edge cases.	
		`I.e.2`: Ensures comprehensive coverage of all relevant input variations (e.g., data types, memory layouts, value ranges).	
		`I.e.3`: Generates inputs that the code under testing is sensitive to.	
	End-to-end Testing	`I.f.1`: Exercises all relevant parts of the code being tested.	
		`I.f.2`: Prevents non-deterministic ("flaky") test results.	
State Modification	State Match	`I.g.1`: Ground truth includes all states achievable after success.	
		`I.g.2`: Checks relevant and irrelevant states for the challenge.	
		`I.g.3`: Ground truth is complex to prevent trivial state modifications.	
Multistep Reasoning	Answer Match	`I.h.1`: Specifies required answer formats in challenge descriptions.	
		`I.h.2`: Minimizes the possibility of success by random guessing.	
	Quality Measure	`I.i.1`: Designs quality metrics that prevent exploitation (e.g., achieving high scores by reward hacking).	

Task Validity		
Aspect	**Question**	**Score**
Tool	`II.1`: Versions of all tools (e.g., Python) are clearly specified.	
	`II.2`: Required API tools are consistently accessible during evaluation.	
	`II.3`: Evaluation process terminates or handles errors appropriately if an API becomes inaccessible.	
Environment	`II.4`: Residual data or state are fully cleared between runs.	
	`II.5`: Agent is completely isolated from any ground truth information.	
	`II.6`: Setup does not change over time (e.g., no live website).	
Reliability	`II.7`: Annotated ground truth is verified for correctness.	
	`II.8`: Each task is verified to be solvable.	
	`II.9`: Benchmark includes an Oracle solver that can automatically solve all challenges.	
	`II.10`: Implementation is free of vulnerabilities that could be exploited to pass evaluations without completing tasks.	

Benchmark Reporting		
Aspect	**Question**	**Score**
Transparency and Validity	`III.1`: Is fully or at least partially open-sourced.	
	`III.2`: Offers an open-source evaluation harness for users.	
	`III.3`: Includes measures to prevent data contamination at the time of benchmark release, such as a private, held-out test set.	
	`III.4`: Includes measures or plans to consistently update challenges over time to avoid overfitting.	
	`III.5`: Clearly states the relationship between the agent capabilities it aims to evaluate and the constructs or outcomes it measures.	
	`III.6`: Clearly states the evaluation subjective of the benchmark (e.g., a model or an agent framework).	
Flaw Mitigation	`III.7`: Describes steps taken to prevent, identify, and correct flaws.	
	`III.8`: Includes qualitative discussions of the potential impact of unavoidable flaws.	
	`III.9`: Includes quantitative analysis to assess the impact of unavoidable flaws (e.g., noise of ground truth).	
Result Interpretation	`III.10`: Reports metrics about statistical significance, such as confidence intervals.	
	`III.11`: Provides guidance on interpreting results with eval flaws.	
	`III.12`: Reports results of non-AI baselines (e.g., human experts).	
	`III.13`: Reports results of trivial agents (e.g., one that does nothing).	