Are LLMs the END of Programming?

Ilan Barr, Nancy Patel, Yifan Yuan

Abstract

This study evaluates the performance of Large Language Models (LLMs) against human coders in solving programming challenges across three programming languages and problem categories using a dataset from LeetCode. We compare five LLMs-GPT-3.5, GPT-4, Claude Haiku, Claude Sonnet, and Claude Opus-focusing on solution acceptance rates, runtime efficiency, memory usage, and error analysis. Preliminary findings show that LLMs generally surpass human coders, particularly after adjusting human performance metrics to account for biases in data collection. The analysis also reveals that problem characteristics such as complexity and description length significantly influence LLM performance. While LLMs demonstrate substantial potential in enhancing programming efficiency, our results suggest a more complementary role alongside human expertise, particularly in complex problem-solving scenarios.

1. INTRODUCTION

The rapid evolution of Large Language Models (LLMs) has brought an onslaught of automation capabilities, from audio transcription, text parsing, and generative features. With such capabilities, these LLMs are rapidly integrating into multiple professions, such as providing sports commentary in real-time, summarizing legal documents, and even creating music. Our project is interested in the integration of LLMs in a different professional landscape: software engineering. Through our study, we aim to delve into the relative strengths and weaknesses of LLMs versus humans and examine the feasibility of LLMs working in conjunction with human software engineers and potentially even substituting them altogether in the future.

Related Work

This March, the entrance of DevinAI shocked the software engineering world [1]. The team behind DevinAI claims it is the "first fully autonomous AI software engineer", capable of building apps end-to-end and autonomously finding and fixing bugs. However, metrics released for DevinAI conclude that it can only solve approximately 14% of the issues it is given. Thus, some critics believe that human-AI pair programming will be the more realistic future of software engineering [2].

"Is GitHub copilot a substitute for human pairprogramming?" investigates the productivity and quality of human-human pair programming versus human-AI programming using GitHub Copilot. Imai finds that although the human-AI pairing generated more lines of code than the human-human pairing in the same amount of time, the code quality was lower [3]. Ma et. al. observe in their paper "Is AI the better programming partner?" that pair programming (both human-human and human-AI) is less productive when there are mismatched expertise levels between the two parties. Thus, the authors hypothesize that, for human-AI pairing to be effective, AIs should be able to adapt to different expertise [4].

To determine the relative strengths and weaknesses of the popular AI Chat-GPT, Nascimento et. al assessed LLMgenerated code versus software-engineer-generated code on popular technical coding interview site, Leetcode. The study finds that GPT-4 outperforms novice programmers, solely when tackling coding problems of easy and medium difficulty. However, the paper did not have enough evidence to assert that GPT-4 surpasses experienced programmers in coding problems of any difficulty (easy, medium, or hard). For their study, Nascimento et. Al focus on 1 LLM (GPT-4), 1 programming language (C++), and 1 question category (Array) [5].

We seek to expand on the empirical study conducted by Nascimento et. al. by testing 5 LLMs, 3 programming languages, and 3 question categories. By exploring how humanauthored versus AI-generated solutions for specific programming paradigms compare against each other, we aim to continue developing the groundwork for more informed human-in-the-loop approaches to software engineering.

2. PRELIMINARIES

Before describing our experiments, we briefly discuss the setting in which we collected all of our human- and LLM-generated code. Our collection process took place between April 1-25, 2024, before GPT-4 rolled out its cross-chat memory control [6]. As a result, we expect that each LLM-generated code snippet we collected is independent of the other LLM-generated code snippets (i.e. each LLM response is unaffected by other responses in separate chats).

3. METHODOLOGY

3.1 Question Collection

Our experiments begin by developing a dataset of 9 questions from LeetCode. We chose LeetCode for our dataset over other coding benchmarks, like SWE-Bench or HumanEval, because the latter datasets do not offer humancompleted solutions (with total acceptance rates) for comparison [10, 11]. For our dataset, we picked three questions each from three of the most popular programming categories, Dynamic Programming (#5, #10, #53), Matrix (#542, #1210, #1970), and Sorting (#217, #630, #2402). Note that many of these questions were "multi-tagged." For example, #1970 is assigned to the Matrix category, but also belongs to the Array, Depth First Search (DFS), Breadth First Search (BFS), Hash Map, and Union Find categories. In total, there is 1 easy-difficulty question, 3 medium-difficulty questions, and 5 hard-difficulty questions. Because Nascimento et. al. found that GPT-4 only outperformed novice programmers on easy and medium-difficulty questions, we are interested in delving deeper into LLM performance on hard-difficulty coding problems. As a result, our question selection is skewed toward harder problems.

3.2 Programming Language Selection

For each question, we collect 3 human submissions and 3 AI-generated codes (for each of our 5 LLMs) in each of the following programming languages: Python, C++, and Go. We chose these languages because we are interested in seeing if LLMs can consistently replicate their performance on languages with different structures, memory capabilities, and programming paradigms. Python is the most popular of the three languages because of its simplicity: it is dynamically typed, has automatic garbage collection, and follows an object-oriented paradigm. C++ is second-most popular: it is object-oriented like Python but does not support automatic garbage collection. Lastly, Go is the least popular: it is statically typed like C++, but unlike C++ and Python, it is a procedural programming language [8].

3.3 Large Language Model Selection

We chose to use the following 5 LLMs: GPT-3.5, GPT-4, Claude Haiku, Claude Sonnet, and Claude Opus. This choice was inspired by the benchmark comparisons released by DevinAI which compares DevinAI against GPT-3.5, GPT-4, and Claude 2 [1]. According to Anthropic (the team behind Claude 2 and 3), the LLMs are ranked as following for HumanEval: Opus (84.9%), Haiku (75.9%), Sonnet (73.0%), GPT-4 (67.0%), and GPT-3.5 (48.1%) [7]. We expect to see similar results in section 4.

3.4 Code Collection

3.4.1 Human-Generated Solutions On LeetCode, there is a "Discussion" section accompanying each problem, where users post their solutions. For our study, we used this section to collect human-generated code using the process outlined as follows. First, we filter the "Discussion" section based on the programming language of interest (Python,

C++, or Go). It is important to note that posts are sorted by rating and views, with the highest-rated and most-viewed posts displayed first. Thus, to minimize any sampling bias, we randomly pick each submission using a random number generator.

3.4.2 AI-Generated Solutions To ensure a robust evaluation of artificial intelligence models, our methodology involved a carefully designed setup for collecting AI-generated solutions. The process is outlined as follows:

- 1. Question Modification: Each problem selected from LeetCode was initially modified to create a novel challenge not directly encountered during the training of the language models. This modification was aimed at assessing the genuine problem-solving abilities of the models under conditions that prevented any benefit from previous exposure to similar problems. If a modification led to all language models (LLMs) consistently failing to pass any tests—indicating an excessively stringent alteration—the original version of the problem was used to maintain a balance of challenge and feasibility.
- 2. **Isolated Problem-Solving Sessions:** Each LLM was tasked with solving these problems in a separate, controlled environment. This isolation ensured that the performance data from one model did not influence another, thereby maintaining the integrity of the experimental conditions.
- 3. Uniform Problem Presentation: The problems were presented to each LLM using a standardized prompt format. As demonstrated in Figure 1, the prompt included a clear articulation of the problem requirements, constraints, and expected outputs, along with a code template for implementation. The standardized prompt helped to ensure that all models were provided with the same level of detail and clarity, minimizing variability in understanding or interpretation that could affect the performance outcomes.

3.5 Research Metrics

In their paper "Perfection Not Required?" Weisz et. al ask software engineers under what conditions they would accept LLM-generated code [9]. The study finds that software engineers tend to accept code through verification, not explanation. In other words, functional metrics were more important to these engineers than non-functional metrics (i.e. explanation or readability).

Thus, we chose to measure human and AI-generated solutions using the following functional metrics: number of passing tests, runtime performance, and memory usage. To test each of these metrics, we take each solution and execute it within a simulated LeetCode environment. Once submitted, the site would provide three pieces of information: the number of test cases passed, the runtime (and the percentage of other submissions it beat), and the memory usage (and the percentage of other submissions it beat).

In some instances, the executed code did not provide any of these metrics because of errors like "Compile Error",

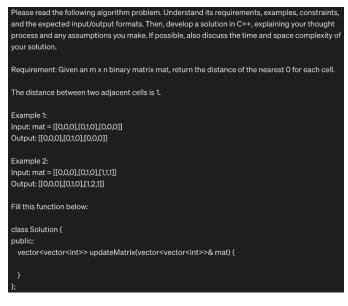


Figure 1: Example Prompt for AI LeetCode Task

"Runtime Error", and "Time Length Exceeded". The one exception is the "Wrong Answer" error, which would offer the number of passing tests, but not runtime performance, and memory usage. A more detailed error analysis is offered in section 4.4.

Lastly, we note that there is potential survivorship bias in the human-generated responses because users only post correct submissions to the "Discussion" section. To mitigate this bias, we collect each problem's acceptance rate to restandardize the percentage of human-generated codes that are accepted by LeetCode.

4. RESULTS

4.1 Overall Performance Evaluation

Across all question categories (Dynamic Programming, Matrix, and Sorting), there are varied success rates for each LLM. Notably, Claude Opus achieved the highest overall solution passing rate at 84.19%, followed by GPT-4 with a 75.44% passing rate as shown in Figure 2.

Further, within each category, performance was significantly influenced by the nature of the selected questions. In each category, questions that had fewer submissions and longer descriptions on LeetCode generally had lower passing rates, suggesting that the complexity of problem descriptions and their popularity among coders might affect LLM performance more than the problem category itself. However, one should note that problem complexity/length are often correlated with problem category on LeetCode. For example, there are more easy-difficulty and lower-complexity problems in the Array category than in the Dynamic Programming category.

4.2 Comparison of LLMs

Across all categories, GPT-4 consistently outperformed other models in terms of runtime, beating 68.56% of the sub-

missions on LeetCode. GPT-4 also showed superior memory management, beating 61.49% of the solutions on LeetCode. In contrast, despite Opus's strong passing rates, it lagged in runtime and memory efficiency, with only 59.85% and 54.34% of submissions beaten, respectively, which is the lowest among all models.

4.3 Human vs. LLM Performance

For the solutions we collected, human participants appeared to perform exceptionally well with a 96.23% passing rate. However, these figures were derived from a pool of already successful solutions, introducing a survivorship bias that inflated human performance metrics.

As mentioned in section 3.5, to address this bias and provide a fair comparison, we recalculated human performance metrics by adjusting for the inherent difficulty of the problems, as reflected by the average passing rates on LeetCode. Upon this adjustment, the real passing rate for human solutions was significantly reduced to 44.81%. This re-calibrated figure starkly contrasts the LLMs' performance; even the lowest-performing LLM, Claude Haiku, demonstrated a solution passing rate of 67.33%, markedly higher than the adjusted human rate.

Furthermore, our experimental results show that AI models substantially outperformed human programmers in terms of runtime and memory efficiency. For instance, GPT-4 consistently exceeded human performance, with its code beating 68.56% of LeetCode submissions in terms of runtime. GPT-4 generated code also beat 61.49% of Leetcode submissions in terms of memory efficiency. In comparison, human solutions only managed to beat 54.98% of submissions in runtime and 53.44% in memory performance on average. Even the least efficient LLM model in these respects, Opus, performed better than human coders, with runtime and memory metrics beating 59.85% and 54.34% of LeetCode submissions, respectively.

Initially, we had planned to conduct a t-test analysis to determine if the mean passing rates of human solutions versus LLM-generated solutions are statistically significant. However, during our data preparation phase, we noted that Leetcode does not release standard deviations for each of its problems' passing rates. Attempting to estimate them using the data from submissions would result in an extremely low standard deviation for human submissions because we are collecting from a (correct) subset of total submissions. Thus, conducting such a test would have meaningless results because of the skewed representation of data. Hence, we compare empirical averages for our analyses instead.

4.4 Error Analysis

Next, we conduct an error analysis, documenting which errors are most frequent among each of the LLMs. Our error analysis reveals that across all LLMs, the most frequent error types are "Wrong Answer" and "Runtime Error". As shown in Figure 2, both GPT-4 and Opus models exhibited a higher proportion of "Wrong Answers" and "Runtime Errors" compared to the other error categories. This observation suggests that the code generated by these models possessed the correct syntax and successfully passed the compilation phase.

		ALL Languages								
			Solution	Accepted				Error		
Category	LLM	Passed Tests	Runtime Performance	Memory Performance	Total Error	Total Wrong Answers	Total Compile Errors	Total Runtime Errors	Total Time Limit Exceeded	Total Memory Limit Exceeded
	GPT3.5 Summary	72.54%	64.31%	56.02%	32	22	4	6	0	0
	GPT4 Summary	75.44%	68.56%	61.49%	23	12	3	5	3	0
	Haiku Summary	67.33%	73.06%	57.43%	33	11	7	9	6	0
ALL	Sonnet Summary	71.74%	68.37%	59.63%	35	22	4	6	2	1
	Opus Summary	84.19%	59.85%	54.34%	19	13	1	4	1	0
	Human Summary	96.23% x 46.57%	54.98%	53.44%	5	2	1	1	1	0
	Cross Model Summary	74.25%	66.83%	57.78%	142	80	19	30	12	1

Figure 2: Summary of Performance Metrics Across ALL Languages and All categories

However, the presence of "Wrong Answers" and "Runtime Errors" indicates the existence of logical inconsistencies or the failure to consider edge cases within the code, leading to the unsuccessful passing of test cases.

In contrast, the Claude Haiku model demonstrated a different error profile, with a higher incidence of "Compile Errors" and "Time Limit Exceeded" (TLE) errors. Compile errors arise when the code violates the syntax or semantics of the programming language, preventing successful compilation. The higher frequency of "compile errors" in the code generated by Claude Haiku suggests potential issues in code generation, such as incorrect syntax, improper variable declarations, or incompatible data types. On the other hand, TLE errors occur when the execution of the code exceeds the allocated time limit. The presence of TLE errors in the Claude Haiku model indicates potential inefficiencies in the generated code, such as sub-optimal algorithms or excessive computational complexity.

Once again, it is worth noting that error analysis for human participants was not applicable in this study, as their solutions were sourced from a pool of previously successful submissions. This approach ensures the correctness and efficiency of human-generated code, serving as a benchmark for evaluating the performance of AI models.

4.5 LLM Performance by Problem Category

The performance of the LLMs varied across the Dynamic Programming (DP), Matrix, and Sorting categories as shown in Figure 3.

In the DP category, all models demonstrated nearly perfect solution acceptance rates, suggesting high competencies in applying DP principles.

The Matrix category, however, proved challenging. The average passing rate for LLMs was 49.05%, with the Haiku model performing the worst at 30.78%. Matrix problems involve complex operations on two-dimensional arrays, indicating limitations in AI models' ability to comprehend and manipulate matrix data structures effectively.

The LLMs generally performed well in the Sorting category, with GPT-4 excelling at an 82.11% passing rate and efficient runtime and memory performance. The strong performance suggests AI models' proficiency in understanding and implementing optimal sorting techniques.

4.6 LLM Performance by Problem Characteristics

Further analysis revealed that Language Models (LLMs) performance was strongly correlated with problem charac-

teristics such as description length and submission rates on LeetCode, rather than problem categories themselves.

Matrix and Sorting problems in our experiment had lower submission rates, as shown in Figure 2 where one of the questions we selected (#1210) for Matrix only had approximately 19,000 submissions. Questions in Matrix and Sorting categories tend to have longer descriptions (over 300 words) as well. Meanwhile, DP problems had higher submissions (9M) and concise descriptions (under 50 words). Despite the inherent complexity of DP problems, LLMs performed better at DP problems, potentially because of the greater availability of training data, higher submission rates, and reduced complexity in understanding problem statements from shorter descriptions.

4.7 LLM Performance by Programming Language

Our research yielded intriguing findings regarding the performance of Large Language Models (LLMs) across different programming languages. As depicted in Tables 3, 4, and 5 in the Appendix, LLMs generally exhibited the better performance in Go compared to Python and C++. Notably, for many questions, the average LLM performance surpassed more than 70% of human solutions on LeetCode in terms of runtime and memory efficiency using Go.

However, while LLMs may have a slight advantage in generating efficient code in Go, their overall performance remains relatively consistent across Python and C++. The similarity in performance across languages suggests that LLMs might have the ability to learn and adapt to the specific syntax, semantics, and best practices of different programming languages.

5. DISCUSSION

5.1 Evaluation of Experimental Design and Data Collection

Our experimental design involved testing 5 Large Language Models (LLMs)— GPT-3.5, GPT-4, Claude Haiku, Claude Sonnet, and Claude Opus— across 3 programming languages and 3 distinct coding problem categories on Leet-Code. This broad spectrum allowed us to assess the robustness and versatility of LLMs in varied coding environments and problem complexities. The strength of this approach lies in its comprehensive coverage, which provides a nuanced understanding of LLM capabilities beyond simple task execution, examining their performance in terms of runtime efficiency, memory usage, and error frequency.



Figure 3: Summary of Performance Metrics Across ALL Languages and Different categories

However, while LeetCode offers a controlled environment to systematically evaluate the coding capabilities of LLMs, it does not fully represent the breadth and depth of challenges faced by software engineers (SWEs) in real-world scenarios. LeetCode problems are typically algorithmic puzzles designed to test specific programming skills and theoretical knowledge, which are only a subset of the skills required in software development. In professional settings, SWEs must also handle tasks such as debugging legacy code, integrating APIs, managing databases, working within complex software architectures, and collaborating in team environments. These tasks often involve dealing with ambiguous requirements, making architectural decisions, and understanding the business logic that goes beyond the scope of algorithmic problem-solving typically found on Leet-Code.

The reliance on LeetCode as a test platform may therefore limit our understanding of how well LLMs can perform in real-world software development tasks. The platform's focus on individual, often isolated challenges may overlook how LLMs handle larger, more integrated projects that require a combination of coding proficiency, system design, and maintenance capabilities. Moreover, the discrete nature of LeetCode problems does not account for aspects of software development like version control, code review, scalability considerations, and security implications, which are crucial for a comprehensive assessment of an SWE's work.

In light of these limitations, future research should consider incorporating more diverse and representative tasks that encompass a wider range of software engineering challenges. This could involve using project-based assessments, integrating tasks from open-source projects, or creating simulated software development environments that reflect the complex nature of modern software projects. Such approaches would provide a more accurate gauge of the potential for LLMs to contribute to or enhance the software development lifecycle, thereby offering deeper insights into their practical utility and limitations in the field of software engineering.

5.2 Future Implications

As we have seen in this study, LLMs can reliably outperform humans on Leetcode in a variety of tasks. LLM development is still in its early stages, and companies are pouring billions of dollars a year into the development of more sophisticated LLMs. We expect the trend will continue and the gap between human and LLM software engineers will continuously expand in the coming years.

While the humans submitting LeetCode problems may not be a fair representation of professional software engineers, as we mentioned in section 5.1, if LLMs have not already reached this professional level, they will soon. There is a lot of uncertainty right now about what the future of software development will look like as LLMs continue to improve in that area. Assuming LLMs can code up to par with a professional software engineer, LLMs will be able to do so at a fraction of the cost. Thus, as we have seen in other industries that have been usurped by technology and automation, the jobs surrounding the space will inevitably change.

5.3 Implications for Programming Education

For over 70 years, the evolution of programming languages has been marked by an ongoing quest to simplify the common problems in programming. From the creation of highlevel programming languages in the 1950s, aimed at bridging the gap between human reasoning and machine operations, to the development of user-friendly programming interfaces in recent decades, there has been a continuous effort to make programming simpler and easier to understand. Despite these advancements, programming remains a skill that requires significant knowledge, logical thinking, and talent.

With the advent of Large Language Models (LLMs) as we studied in this experiment, we may be witnessing a pivotal shift. Unlike traditional programming, where mastery over syntax and semantics is essential, LLMs allow individuals to engage in "conversational programming." With this practice, users can code in their spoken language, with LLMs then translating those instructions into functional code. Our study already highlights the vast capabilities of these existing models to generate solutions for complex problems with efficiency and accuracy surpassing human performance.

Given these developments and studies, a fundamental question arises: is it still necessary to teach traditional programming, or should educational systems shift towards teaching students how to effectively interact with LLMs like the GPTs? As LLMs advance rapidly, fueled by the significant investments and research, driven by enormous financial benefits mentioned in section 5.2, traditional coding tasks that involve writing detailed lines of code might become less prevalent.

However, this does not render the teaching of programming obsolete. Instead, it suggests a transformation in the curriculum. Here, we suggest that future programming education should focus more on the following areas:

- AI Literacy: Understanding how AI models work, including their strengths, limitations, and ethical implications. As LLMs become integral tools in programming, education in these technologies must go beyond mere usage instructions. It should aim to build a deep understanding of how AI works, its limitations, and its broader impacts. This will ensure that future professionals are not only proficient in using AI but are also capable of critically assessing, managing, and directing these tools responsibly and ethically in the evolving landscape of software development.
- **Problem-Solving and Design Thinking:** In an era where AI can handle an increasing array of programming tasks, the human role in software development is evolving towards more complex problem formulation and system design. This shift necessitates a deeper focus on problemsolving and design-thinking skills within programming education, which are crucial for leveraging AI capabilities effectively.
- Human-AI Collaboration: As AI technologies become integral to software development processes, the ability to manage projects that leverage both human and AI contributors is becoming increasingly important. Human-AI collaboration is not merely about using AI as a tool but integrating AI into the software development team in a way that amplifies both the AI's capabilities and the human team's skills.

6. CONCLUSION

The results of this study on the application of Large Language Models (LLMs) in programming present a transformative vision for the future of software development and education. While our experiments reveal that LLMs can outperform human developers in efficiency and problemsolving within the constrained environment of LeetCode, these results do not immediately translate to a comprehensive takeover of programming jobs by AI. Instead, the results highlight the potential for AI to augment human abilities, particularly in well-defined problem spaces, where the problem descriptions are concise and straightforward. As AI continues to evolve and as investment in this technology grows, the role of software developers is likely to shift towards more complex and creative problem-solving tasks. Education systems should strategically plan for the upcoming change within the next five to ten years, emphasizing the development of skills that enable effective human-AI collaboration. Classes, like CSE518A: Human-in-the-Loop Computation at Washington University in St. Louis, are great examples of such courses. Ultimately, as the landscape of technology progresses, the fusion of human ingenuity with advanced AI capabilities may unlock new realms of possibility in software development and beyond.

7. REFERENCES

[1]https://www.cognition-labs.com/introducing-devin

[2]https://societysbackend.com/p/devin-has-exposed-software-engineers

[3]https://dl.acm.org/doi/abs/10.1145/3510454.3522684

[4]https://arxiv.org/abs/2306.05153

[5]https://arxiv.org/abs/2305.1183

[6]https://openai.com/blog/memory-and-new-controlsfor-chatgpt

[7]https://www.vellum.ai/blog/llm-benchmarksoverview-limits-and-model-comparison

[8]https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/

[9]https://arxiv.org/abs/2104.03820

[10]https://www.swebench.com/

[11]https://paperswithcode.com/dataset/humaneval

8. APPENDIX

Refer to the next few pages for more information on our full data collection and example LLM-generated codes.

_				Python3					C++ Solidate Accepted				GO		
	Question#		Number of Tests	Passed Texts	Buttime Performance	Allage Cade	Error Type	Passed Tests		Accepted Memory Performance	Error Type	Passed Tests	Buttine Performance		Error Type
907 6	CONTRACT	GPT3.5	210	210(100%)	Beats 52.90%	Beats 77.17%	NA NA	210(100%)	Beats \$1,38%	Beats 29,41%	NA NA	210(100%)	Beats 05.40%	Basis 02.70%	NA
		GPT3.5	210	210(100%)	Beats 41.64%	Beats 37,99%	NA	210(100%)	Beats 50,15%	Beats 67.10%	NA	210(100%)	Beats 77.22%	Beats 00.32%	NA
		GPT3.5	210	210(100%)	Beats 19.32%	Beats 37.99%	NA	210(100%)	Beats 70.09%	Beats 67.10%	NA	210(100%)	Beats 03.37%	Beats 02.70%	NA
		GPT3.5 Summary	210	100%	AVG 37.98%	AVG 51.05%	0 Error	100%	AVG 59.21%	AVG 54.54%	0 Error	100%	AVG 82.00%	AVG 84.62%	0 Error
		GPT4	210	210(100%)	Beats 37.69%	Beats 63.64%	NA	210(100%)	Beats 56, 19%	Beats 92,92%	NA	210(100%)	Beats 92.10%	Beats 90.00%	NA
		GPT4	210	210(100%)	Beats 37.09%	Beats 10.55%	NA	210(100%)	Gents 00.29%	Beats 67.20%	NA	210(100%)	Beats 87.80%	Beats 92.99%	NA
		GPT4	210	210(100%)	Beats 37.09%	Beats 37.00%	NA	210(100%)	Beats 74.19%	Beats 92.93%	NA	210(100%)	Beats 93.20%	Beats 70.72%	NA
		GPT4 Summary	210	100%	AVG 37.69%	AVG 39.36%	0 Error	100%	AVG 72.89%	AVG 04.37%	0 Error	100%	AVG 91.00%	AVG 89.52%	0 Error
		Claude Halku	210	210(100%)	Beats 40.33%	Beats 10.50%	NA	210(100%)	Geats 31.57%	Beats 67.10%	NA	210(100%)	Beats 91.53%	Beats 69.19%	NA
		Claude Haiku	210	210(100%)	Beats 37.00%	Beats 37.99%	NA	210(100%)	Seats 31.57%	Beats 67.10%	NA	210(100%)	Beats 73.07%	Beats 49.53%	NA
52	Medium.	Claude Halku	210	210(100%)	Beats 18.40%	Beats 37.99%	NA	210(100%)	Seats 97.00%	Beats 29.41%	NA	210(100%)	Baats 73.87%	Beats 55.92%	NA
	50.7%	Haiku Summary	210	100%	AVG 32.22%	AVG 30.65%	0 Error	100%	AVG 53.36%	AVG 51.54%	0 Error	100%	AVG 79.70%	AVG 58.21%	0 Error
Ac	Acceptance	Claude Sonnet	210	210(100%)	Beats 5.98%	Beats 16.00%	NA	210(100%)	Beats 5.05%	Basts 9.92%	NA	210(100%)	Beats 10.01%	Beats 62.99%	NA
	Rate	Claude Sonnet	210	210(100%)	Beats 16.70%	Beats 38.02%	NA	210(100%)	Sents 31.64%	Beats 29.67%	NA	210(100%)	Beats 75.89%	Beats 24.57%	NA
		Claude Sonnet	210	210(100%)	Beats 7.12%	Beats 38.02%	NA	210(100%)	Beats 21.64%	Beats 29.67%	NA	210(100%)	Beats 70.12%	Beats 70.54%	NA
		Sonnet Summary	210	100%	AVG 9.90%	AVG 30.89%	0 Error	100%	AVG 22.70%	AVG 23.09%	0 Error	100%	AVG 57.34%	AVG 61.37%	0 Error
		Claude Opus	210	210(100%)	Beats 11.74%	Beats 16.63%	NA	210(100%)	Beats 70.12%	Beats 29.67%	NA	210(100%)	Beats 53.61%	Beats 99.53%	NA
		Claude Opus	210	210(100%)	Beats 32.35%	Beats 16.63%	NA	210(100%)	Beats 10.99%	Beats 29.67%	NA	210(100%)	Beats 64.10%	Beats 82.99%	NA
		Claude Opus	210	210(100%)	Beats 15.30%	Beats 16.63%		210(100%)	Beats 51,49%	Beats 29.67%		210(100%)	Beats 92.32%	Beats 56.83%	
		Opus Summary	210	100%	AVG 19.80%	AVG 16.63%	0 Error	100%	AVG 44.20%	AVG 29.67%	0 Error	100%	AVG 76.70%	AVG 78.70%	0 Error
		Human 1	210	210(100%)	Beats 43.97% Beats 92.70%	Beats 33.03% Beats 63.09%	NA	210(100%)	Beats 65.40% Beats 60.95%	Beats 29.95% Beats 67.42%	NA	210(100%)	Basts 91.85% Basts 00.35%	Beats 63.47% Beats 93.72%	NA
		Human 2 Human 3	210	210(100%)	Beats 92.70% Beats 14.50%	Beats 03.09% Beats 10.00%	NA	210(100%)	Beats 60.95% Beats 70.50%	Beats 67.42% Beats 29.95%	NA	210(100%)	Beats 00.35% Beats 45.03%	Baats 93.72% Baats 57.61%	NA
		Human 3 Human Summary					0 Error				0 Error				0 Emp
		Profiles Summary	210	100%	AVG 50.44%	AVG 39.44%	O LINDA	100%	AVG 65.00%	AVG 42.44%	O Linter	100%	AVG 74.00%	AVG 71.00%	06/10
		GPT3.5	542	142(100%)	Beats 07.97%	Beats 04.54%	NA	142(100%)	Eests 79.57%	Beats 93.14%	NA	142(100%)	Beats 09.11%	Beats 40.92%	NA
		GPT3.5	542	142(100%)	Beats 07.21%	Beats 50.25%	NA	142(100%)	Beats 29.02%	Beats 37.00%	NA	142(100%)	Bests 09.11%	Beats 60.21%	NA
		GPT3.5	542	142(100%)	Beats 03.72%	Beats 04,54%	NA	142(100%)	Beats 70.35%	Beats 93.14%	NA	142(100%)	Deate 100.00%	Beats 40.92%	NA
		GPT3.5 Summary	542	100%	AVG 79.64%	AVG 73.11%	0 Error	100%	AVG 03.25%	AVG 74.69%	0 Error	100%	AVG 78.41%	AVG 50.02%	0 Erro
		GPT4	542	142(100%)	Beats 06.04%	Beats 04.53%	NA	142(100%)	Beats 96.77%	Beats 02.74%	NA	142(100%)	Basts 09.42%	Beats 40.93%	NA
		GPT4	542	0(0%)	NA	NA	Destina Dese	142(100%)	Deats 04.90%	Beats 02.74%	NA	142(100%)	Beats 02.44%	Beats 67.30%	NA
		GPT4	542	142(100%)	Basts 82.49%	Beats 84 53%	NA	542(100%)	Gents 75.30%	Beats 73 43%	NA	142(100%)	Basis 09.42%	Dants 67,36%	NA
		GPT4 Summary	542	67%	AVG 04 57%	AVG 64.53%	1 Error	500%	AVG 05 60%	AVG 79.64%	0 Error	100%	AVG 74.09%	AVG SELSON	0 Emp
		Claude Halku	542	142(100%)	Beats 34.74%	Beats 51,71%	NA	142(100%)	Beats 34,70%	Beats 33.62%	NA	142(100%)	Basts 45.46%	Beats 15,73%	NA
		Claude Halku	542	142(100%)	Beats 28.72%	Beats 11,71%	NA	142(100%)	Gents 19.00%	Beats 41.55%	NA	142(100%)	Beats 23.07%	Beats 20.32%	NA
		Claude Halku	542	142(100%)	Beats 72.10%	Beats 50.27%	NA	542(100%)	Beats 70.49%	Beats 73 60%	NA	00%)	NA	NA	Consider
	Medium,	Haiku Summary	542	100%	AVG 45.19%	AVG 20.56%	0 Error	100%	AVG 41.04%	AVG 49.62%	0 Error	67%	AVG 34.27%	AVG 18.03%	1 Erro
	33.8%	Claude Sonnet	542	142(100%)	Beats 72.40%	Beats \$1.55%	NA	142(100%)	Beats 79,49%	Beats 93.15%	NA	142(100%)	Beats 09.30%	Beats (6.55%)	NA
Ac	cceptance	Claude Sonnet	542	142(100%)	Beats 74,00%	Beats 01.55%	NA	126(99%)	NA	NA	Winner Annuar	142(100%)	Beats 09.30%	Beats 65.55%	NA
	Rate	Claude Sonnet	542	142(100%)	Beats 00.12%	Beats 50.35%	NA	142(100%)	Gents 36.70%	Beats 37.00%	NA	142(100%)	Beats 22.37%	Beats 9.04%	NA
		Screet Summary	542	100%	AVG 77.53%	AVG 73.15%	0 Error	99%	AVG 58.13%	AVG 65.38%	1 Error	100%	AVG 53.00%	AVG 46.90%	0 Emp
		Claude Opus	142	142(100%)	Beats 63.01%	Beats 50.35%	NA	142(100%)	Beats 57,13%	Beats 21.40%	NA	142(100%)	Basts 03.01%	Banta 65,55%	NA
		Claude Opus	142	142(100%)	Beats 06.12%	Beats 01.55%	NA	142(100%)	Gents 36,70%	Beats 34.02%	NA	142(100%)	Seats 100.00%	Baats 65.55%	NA
		Claude Opus	542	142(100%)	Beats 92.94%	Beats 04.55%	NA	142(100%)	Beats 35.90%	Beats 37.00%	NA	142(100%)	Beats 03.04%	Beats 65.55%	NA
		Opus Summary	542	100%	AVG 02.30%	AVG 73.15%	0 Error	100%	AVG 43.29%	AVG 21.01%	0 Error	100%	AVG 89.23%	AVG 65.55%	0 Emp
		Human 1	542	142(100%)	Beats \$4,02%	Beats 50.25%	NA	142(100%)	NA	NA	Tires Link Escended	142(100%)	Beats 70.90%	Baats 39,99%	NA
		Human 2	542	142(100%)	Beats 00.04%	Beats 04.59%	NA	142(100%)	Gents 61.40%	Beasts 51.27%	NA	142(100%)	Seats 100.00%	Beats 55.66%	NA
		Human 3	542	142(100%)	Beats 32.5%	Beats 7.21%	NA	142(100%)	Gents 41,44%	Beasts 37.09%	NA	142(100%)	Beats 51.57%	Baats 39.99%	NA
		Human Summary	210	100%	AVG 44.45%	AVG 47.35%	0 Error	100%	AVG 51.45%	AVG 44.48%	1 Error	100%	AVG 74.10%	AVG 45.22%	0 Em
		Cross Model Summary	142	AVG Passing Rate \$3.40	% AVG 73.06%	AVG 00.10%	1 Emor(x)	AVG Passing Rate 99.00	% AVG SEX0%	AVG 60.07%	1 Emar(a)	AVG Passing Rate 93.4	FS AVG 66.12%	AVG 47.83%	1 Emp
		GPT3.5	356	256(100%)	Beats 55.40%	Beats 74.00%	NA	256(100%)	Beats 68.21%	Beats 53.00%	NA	356(100%)	Sents 100.00%	Beats 47.69%	NA
		GPT3.5	356	356(100%)	Beats 09.07%	Beats 95.73%	NA	356(100%)	Beats 68.21%	Beats 38.03%	NA	356(100%)	Beats 71.00%	Beats 47.69%	NA
		GPT3.5	356	356(100%)	Beats 26.22%	Beats 74.00%	NA	356(100%)	Beats 68.21%	Beats 45.73%	NA	356(100%)	Beats 64.30%	Beats 54.87%	NA
		GPT3.5 Summary	356	100%	AVG 50.45%	AVG 01.20%	0 Error	100%	AVG 01.21%	AVG 45.88%	0 Error	100%	AVG 78.40%	AVG 50.00%	0 En
1	10 Hard,	GPT4	356	356(100%)	Beats 42.44%	Beats 48.90%	NA	256(100%)	Beats 29.81%	Beats 30.13%	NA	356(100%)	Beats 64.47%	Beats 27.16%	NA
	28.1%	GPT4	356	356(100%)	Reats 05.30%	Beats 48.90%	NA	256(100%)	Beats 30.27%	Beats 22.39%	NA	356(100%)	Beats 31.22%	Beats 15.74%	NA
Ac	coeptance	GPT4	356	256(100%)	Beats 87.09%	Beats 95.04%	NA	256(100%)	Beats 68.43%	Beats 22.39%	NA	356(100%)	Beats 70.01%	Beats 46.70%	NA
R	ate(Note:	GPT4 Summary	356	100%	AVG 71.90%	AVG 61.41%	0 Error	100%	AVG 42.04%	AVG 27.64%	0 Error	100%	AVG 55.50%	AVG 29.07%	0 En
	For	Claude Halku	356	356(100%)	Beats 60.33%	Beats 40.00%	NA	356(100%)	Beats 30.27%	Beats 38.20%	NA	356(100%)	Beats 100.00%	Beats 50.66%	NA
	Problem	Claude Halku	356	356(100%)	Beats 51.01%	Beats 75.00%	NA	356(100%)	Beats 68.42%	Beats 54.04%	NA	356(100%)	Beats 100.00%	Beats 48.51%	NA
	10, about	Claude Haiku	356	356(100%)	Beats 96.05%	Beats 40.00%	NA	356(100%)	Beats 68.42%	Beats 38.20%	NA	356(100%)	Beats 64.36%	Beats 48.51%	NA
	70% of	Haiku Summary	356	100%	AVG 09.40%	AVG 57.59%	0 Error	100%	AVG 55.71%	AVG 43.48%	0 Error	100%	AVG 88.12%	AVG 51,20%	0 En
	ubmisson	Claude Sonnet	356	289(81%)	NA	NA	Wrong Antwer	356(100%)	Bast. 100.00%	Beats 51.09%	NA	356(100%)	Beats 100.00%	Beats 47.33%	NA
	ave 0 ms	Claude Sonnet	356	356(100%)	Beats 38.63%	Beats 74.00%	NA	356(100%)	Bast. 100.00%	Beats 22.15%	NA	356(100%)	Beats 24.60%	Beats 90.00%	NA
	for C++. For GO.	Claude Sonnet	356	287(81%)	NA	NA	Wrong Antwer	356(100%)	Basis 100.00%	Beats 38.36%	NA	356(100%)	Beats 05.14%	Beats 47.33%	NA
	bout 30%	Sonnet Summary	256	67%	AVG 38.63%	AVG 74.08%	2 Error	100%	AVG 100.00%	AVG 38.20%	0 Error	100%	AVG 63.27%	AVG 61.50%	0 En
	people	Claude Opus	356	356(100%)	Beats 35.34%	Beats 95.01%	NA	256(100%)	Bass 100.00%	Beats 30.17%	NA	356(100%)	Beats 65.14%	Beats 47.33%	NA
	have	Claude Opus	356	356(100%)	Beats 32.17%	Beats 48.82%	NA	256(100%)	Beats 68.00%	Beats 30.17%	NA	356(100%)	Beats 65.14%	Beats 21.04%	NA
	- CREWINE	Claude Opus	356	356(100%)	Beats 46.53%	Beats 95.01%	NA	0(0%)	NA AVG 84.02%	NA	Complex Error	356(100%)	Bests 05.14%	Beats 53.44% AVG 43.94%	NA
	domination 1						0 Enter			AVG 30.17%	1 Enter	100%			0 Em
	ubmission of 0 ms)	Opus Summary Human 1	356	100% 256(100%)	Beats 70.02%	AVG 80.01%	NA NA	256(102%)	Beats 14,50%	Beats 12.57%	NA	255(99.7%)	NA	NA	C BITS

Figure 4: Appendix for All DP Questions Collected

		nation# LLM Number of Teats		Pythan3				C++				GO				
Calegory	Quastor#			Passed Texts	Bolution Accepted Error Nonline Performance Error Type		Passed Texts	Solution Rantime Performance	Solution Accepted time Performance Memory Performance		Passed Tests	Solution Accepted Runtime Performance: Nerrory Performance		Error Type		
	LLM DP Su			AVG Papatres Rate 98,93%	AVG 51.695	AVG 57.12%	3 Error(a)	AVC Devolves Note 07 795	AVC 50 68%	AVG 48,795	Error Type 2 Error(s)	AVG Passing Rate 97,80%	AVG 71.20%	AVG 58.625	1 Ferraria)	
		GPT3.5	42	0(0%)	NA	NA	Runime Error	D(0%)	NA	NA	Comple Error	42(100%)	Beats 85.40%	Beats 82.78%	NA	
		GPT3.5	42	0(0%)	NA	NA		0(0%)	NA	NA		42(100%)	Beats 77.23%	Beals 88.32%	NA	
		GPT3.5	42	0(0%)	NA	NA	Plantime Erman	0(0%)	NA	NA	Comple Error	42(100%)	Beats 83.37%	Beats 82.78%	NA	
		GP13.5 Summary	42	0%	NA	NA	3 Error	0%	NA	NA	3 Error	100%	AVG 82.00%	AVG 84.83%	0 Error	
		GPT4	42	42(100%)	Beats 100.00%	Beats 33.33%	NA	16(38%)	NA	NA	Wrong Arouset	0(0%)	NA	NA	Numbers Error	
		GPT4	42	42(100%)	Beats 100.00%	Beats 33.33%	NA	16(38%)	NA	NA	Wrong Anoser	0(0%)	NA	NA	Nuntime Error	
		GPT4	42	42(100%)	Beerbs 100.00%	Beats 33.33%	NA	16(38%)	NA	NA	Wrong Arouset	16(38%)	NA	NA	Writing American	
		GP14 Summary Claude Maku	42 42	100%	AVG 100.00% NA	AVG 33.33% NA	0 Error	38%	NA NA	NA NA	3 Error	13% 0(0%)	NA NA	NA NA	3 Error	
		Claude Haiku	42	4(10%)	NA	NA		4(10%)	NA	NA		4(10%)	NA	NA		
		Claude Halky	42	1(2%)	NA	NA		1(2%)	NA	NA		1(2%)	NA	NA		
	1210 Hard,	Halku Summary	42	4%	NA	NA	3 Error	4%	NA	NA	3 Error	4%	NA	NA	3 Error	
	49.5%	Claude Sonnel	42	0(0%)	NA	NA	Memory Linit Exceeded	0(0%)	NA	NA	Comple Error	0(0%)	NA	NA	Complia Error	
	Acceptance Rate	Claude Sonnel	42	0(0%)	NA	NA		17(40%)	NA	NA	Wrong Aroneet	0(0%)	NA	NA	Comple Error	
	Padite	Claude Sonnel	42	0(0%)	NA	NA	Natime Error	16(38%)	NA	NA	Wrong Anoser	16(38%)	NA	NA	Wrong American	
		Sonnel Summary	42	0%	NA	NA	3 Error	26%	NA	NA	3 Error	13%	NA	NA	3 Error	
		Claude Opus	42	18(38%)	NA	NA	Wrong Areaser	16(38%)	NA	NA	Wrong Anoser	16(38%)	NA	NA	Wrong Anavar	
		Claude Opus	42	0(0%)	NA	NA NA		1(2%)	NA NA	NA NA	Wrong Anoser	0(0%)	NA NA	NA	Nuntime Error	
		Claude Opus		18(38%)			Wrong Areceset	18(38%)			Wrong Anonear	16(38%)		NA	Writing Arussia	
		Opus Summary Human 1	42	25%	NA Rest State	NA Deats 68,13%	3 Error NA	28%	NA Resta \$2.000	NA Rests 72 218	3 Error NA	25%	NA Route 52,612	NA Resta 00 535	3 Error NA	
		Human 2	42	42(100%) 42(100%)	Beats 85.48% Beats 27.42%	Beats 27 A2%	NA	42(100%) 42(100%)	Beats 83.93% Beats 50.00%	Beats 73.21% Beats 64.29%	NA	42(100%) 42(100%)	Beats 53.81% Beats 84.18%	Beats 99.53% Beats 82.99%	NA	
		Human 3	42	42(100%) 42(100%)	Beats 51,61%	Beats 27.42% Beats 75.81%	NA	42(100%) 42(100%)	Beats 50.00% Beats 39.29%	Beats 54.25% Beats 5.37%	NA	42(100%) 42(100%)	Beats 94.16% Beats 92.32%	Beals 52.99% Beals 56.83%	NA	
		Human Summary	42	100%	AVG 54.84%	AVG 58.45%	0 Error	AVG 100.00%	AVG 57.74%	AVG 47.82%	0 Errora	100%	AVG 78,70%	AVG 79.78%	0 Error	
		Cross Model Suremany	42	AVG Passing Rule 25 80%	AVG 100.00%	AVG 33, 33%	12 Error(s)	AVG Passing Rule 18 80%	NA	NA	15 Error(s)	AVG Passing Res 31 00%	AVG 82.00%	AVG 84,835	12 Error(s)	
		GPT3.5	118	0(0%)	NA	NA	Wronet Argement	118(100%)	Beats 5.11%	Beats 15.02%	NA	2(2%)	NA	NA	Where American	
		GPT3.5	118	4(3%)	NA	NA	Wrong Areaser	0(0%)	NA	NA	Wrong Arouser	0(0%)	NA	NA	Writing American	
		GPT3.5	118	1(1%)	NA	NA	Wrong Areineet	0(0%)	NA	NA	Comple Error	0(0%)	NA	NA	Writing American	
		GPT3.5 Summary	118	1%	NA	NA	3 Error	33%	AVG 5.11%	AVG 15.02%	2 Error	1%	NA	NA	3 Error	
		GPT4	118	D(0%)	NA	NA	Wrong Areaset	0(0%)	NA	NA	Wrang Arbonet	0(0%)	NA	NA	Writing American	
		GP14	118	48(39%)	NA	NA	Wrong Aresser	118(100%)	Beats 14.33%	Beats 20.60%	NA	0(0%)	NA	NA	Wrong American	
		GPT4	118	0(0%)	NA	NA	Numbre Error	1(1%)	NA	NA	Wrong Anoser	0(0%)	NA	NA	Complia Error	
		GP14 Summary	118	13%	NA	NA	3 Error	34%	AVG 14.33%	AVG 20.60%	2 Error	0%	NA	NA	3 Error	
		Claude Haiku Claude Haiku	118	D(0%) 20(17%)	NA	NA NA		D(0%)	NA	NA		0(0%)	NA.	NA		
		Claude Halku	118	20(17%) 20(17%)	NA	NA.		20(17%) 0(0%)	NA	NA		0(0%)	NA	NA	Numbers Error	
	1970 Hard,	Halku Summary	118	20(17%) 11%	NA	NA	3 Error	6%	NA	NA	3 Error	0%	NA	NA	3 Error	
Matrix	62.0%	Claude Sonnel	118	0(0%)	NA	NA	Warner Arrenar	1(1%)	NA	NA	Viewer Arrenar	94(80%)	NA	NA	Three Livest Discourses	
in anna	Acceptance	Claude Sonnel	118	0(0%)	NA	NA		0(0%)	NA	NA		93(80%)	NA	NA		
	Rate	Claude Sonnel	118	0(0%)	NA	NA		0(0%)	NA	NA		5(4%)	NA	NA		
		Sonnet Summary	118	0%	NA	NA	3 Error	0%	NA	NA	3 Error	55%	NA	NA	3 Error	
		Claude Opus	118	118(100%)	Beats 33.48%	Beats 63.04%	NA	118(100%)	Beats 27.63%	Beats 30.93%	NA	118(100%)	Beats 20.00%	Beats 20.00%	NA	
		Claude Opus	118	118(100%)	Beats 31.74%	Beats 72.17%	NA	118(100%)	Beats 40.84%	Beats 29.43%	NA	118(100%)	Beats 40.00%	Beats 20.00%	NA	
		Claude Opus	118	118(100%)	Beats 38.52%	Bearts 63.04%	NA	118(100%)	Beals 54.05%	Beats 28.43%	NA	118(100%)	Beats 40.00%	Beals 20.00%	NA	
		Opus Summary	118	100%	AVG 33.91%	AVG 88.08%	0 Error	100%	AVG 40.84%	AVG 28.93%	0 Error	100%	AVG 33.33%	AVG 20.00%	0 Error	
		Human 1	118	118(100%)	Beats 32.02%	Beats 77.19%	NA	118(100%)	Beats 54.03%	Beats 32.24%	NA	118(100%)	Beats 20.00%	Beals 20.00%	NA	
		Human 2 Human 3	118	118(100%) 118(100%)	Beats 80.7%	Beats 71.93%	NA NA	118(100%) 118(100%)	Beats 38.12%	Beats 34.03%	NA	118(100%) 118(100%)	Beats 20.00%	Beals 60.00%	NA	
		Human Summary	118	118(100%) 100%	Beats 73.25% AVG 61.99%	Beats 39.91% AVG 83.01%	0 Error	100%	Beats 91.94% AVG 60.70%	Beats 87.46% AVG 51.24%	0 Error	115(100%)	Beats 20.00% AVG 20.00%	Beats 20.00% AVG 33.33%	0 Error	
		Cross Model Suremany	118	AUG Passing Nata 25 COS	AVG 33 915	AVG 88 DRS	12 Employ	AUG Personal Nate 34 Arris	AVG 20.05%	AVG 21 52%	10 Emprilia)	AVG Passing Nate 31 208	ALVE ST STR	AVG 20.007	12 Erroriel	
		GPT3.5	50	50(100%)	Beats 78.44%	Beats 72.10%	NA	50(100%)	Beats 71.88%	Beats 82.28%	NA	50(100%)	Beats 67.06%	Beats 8.63%	NA	
		GPT3.5	50	50(100%)	Beats 48.37%	Beats 38.59%	NA	50(100%)	Beats 62.58%	Beats 78.27%	NA	50(100%)	Beats 44.31%	Beats 19.61%	NA	
		GPT3.5	50	0(0%)	NA	NA	Runitme Ernar	50(100%)	Beats 45.84%	Beats 76.27%	NA	50(100%)	Beats 97.85%	Beats 60.78%	NA	
													AVC RD R75		0 Error	
		GP13.5 Summary	50	87%	AVG 63.41%	AVG 54.35%	1 Error	100%	AVG 60.09%	AVG 78.27%	0 Error	100%		AVG 29.87%		
		GP14	50	50(100%)	Beats 73.00%	Beats 72.87%	NA	50(100%)	Beats 66.74%	Beats 76.48%	NA	50(100%)	Beats 84.44%	Beals 75.88%	NA	
		GP14 GP14	50 50	50(100%) 50(100%)	Beats 73.00% Beats 88.80%	Beats 72.87% Beats 59.43%	NA NA	50(100%) 50(100%)	Beats 68.74% Beats 74.71%	Beats 76.48% Beats 82.37%	NA NA	50(100%) 50(100%)	Beats 84.44% Beats 78.21%	Beals 75.88% Beals 85.21%	NA NA	
		GP14 GP14 GP14	50 50 50	50(100%) 50(100%) 50(100%)	Beats 73.00% Beats 66.80% Beats 60.14%	Beats 72.87% Beats 59.43% Beats 59.43%	NA NA NA	50(100%) 50(100%) 50(100%)	Beats 68.74% Beats 74.71% Beats 87.82%	Beats 76.48% Beats 82.37% Beats 82.37%	NA NA NA	50(100%) 50(100%) 50(100%)	Beats 84.44% Beats 78.21% Beats 33.48%	Beals 75.88% Beals 85.21% Beals 27.63%	NA NA NA	
		GP14 GP14 GP14 GP14 Summery	50 50 50 50	50(100%) 50(100%) 50(100%) 50(100%) 100%	Beats 73.00% Beats 86.80% Beats 80.14% AVG 68.65%	Beats 72.87% Beats 59.43% Beats 59.43% AvG 63.91%	NA NA NA 0 Error	50(100%) 50(100%) 50(100%) 100%	Beats 68.74% Beats 74.71% Beats 87.82% AVG 78.42%	Beats 76.48% Beats 82.37% Beats 82.37% AVG 80.41%	NA NA	50(100%) 50(100%) 50(100%) 100%	Beats 84.44% Beats 78.21% Beats 33.46% AVG 65.37%	Beats 75.88% Beats 85.21% Beats 27.63% AVG 82.91%	NA NA NA 0 Error	
		GP14 GP14 GP14 GP14 Summary Chaude Halku	50 50 50 50 50	50(100%) 50(100%) 50(100%) 100% 50(100%)	Beats 73.00% Beats 86.80% Beats 80.14% AVG 88.85% Beats 90.75%	Beats 72.87% Beats 50.43% Beats 50.43% AVG 83.91% Beats 50.12%	NA NA NA 0 Emor NA	50(100%) 50(100%) 50(100%) 100% 11(22%)	Beats 88.74% Beats 74.71% Beats 87.82% AVG 78.42% NA	Beats 76.48% Beats 82.37% Beats 82.37% AVG 80.41% NA	NA NA NA	50(100%) 50(100%) 50(100%) 100% 50(100%)	Beats 84.44% Beats 78.21% Beats 33.46% AVG 85.37% Beats 50.59%	Beats 75.88% Beats 85.21% Beats 27.63% AVG 62.91% Beats 27.45%	NA NA NA O Error NA	
	542	GP14 GP14 GP14 GP14 Summery Checke Helku Checke Helku	50 50 50 50 50 50 50	50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%)	Beats 73.00% Beats 68.80% Beats 60.14% AVG 66.65% Beats 90.75% Beats 90.45%	Beats 72.87% Beats 59.43% Beats 59.43% AVG 63.91% Beats 59.12% Beats 59.12%	NA NA NA O Emor NA NA	50(102%) 50(102%) 50(102%) 100% 11(22%) 11(22%)	Beats 66.74% Beats 74.71% Beats 87.82% AVG 78.42% NA NA	Beats 78.48% Beats 82.37% Beats 82.37% AVG 80.41% NA NA	NA NA NA O Error Runtime Error Runtime Error	50(100%) 50(100%) 50(100%) 100% 50(100%) 50(100%)	Baata 84.44% Baata 78.21% Baata 33.48% AVG 65.37% Baata 50.59% Baata 50.59%	Beats 75.88% Beats 85.21% Beats 27.63% AVG 82.91% Beats 27.45% Beats 85.88%	NA NA O Error NA NA	
	542 Medium	GP14 GP14 GP14 GP14 Summary Claude Halku Claude Halku Claude Halku	50 50 50 50 50 50 50	50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%)	Beets 73.00% Beets 88.80% Beets 80.14% AVG 66.65% Beets 90.75% Beets 90.75% Beets 91.54%	Bantis 72.87% Bantis 59.43% Bantis 59.43% AVG 63.91% Bantis 59.12% Bantis 59.12% Bantis 59.12%	NA NA O Emor NA NA NA	50(100%) 50(100%) 50(100%) 100% 11(22%) 11(22%) 50(100%)	Beets 68,74% Beets 74,71% Beets 87,82% AVG 78,42% NA NA Beets 85,94%	Beats 76.48% Beats 82.37% Beats 82.37% AVG 80.41% NA NA Beats 94.40%	NA NA O Error Huntere Error Huntere Error NA	50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%)	Basta 84.44% Basta 78.21% Basta 33.48% AVG 65.37% Basta 50.59% Basta 98.04% Basta 98.08%	Beats 75.88% Beats 85.21% Beats 27.63% AVG 62.91% Beats 27.45% Beats 85.88% Beats 60.78%	NA NA NA O Error NA NA NA	
	542 Medium, 48.4%	GP14 GP14 GP14 GP14 Summery Checke Helku Checke Helku	50 50 50 50 50 50 50 50 50	50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%)	Beats 73.00% Beats 68.80% Beats 60.14% AVG 66.85% Beats 90.75% Beats 90.45% Beats 91.54% AVG 90.91%	Beats 72.87% Beats 59.43% Beats 59.43% AVG 63.91% Beats 59.12% Beats 59.12%	NA NA NA O Emor NA NA	50(100%) 50(100%) 50(100%) 100% 11(22%) 11(22%) 50(100%) 48%	Beats 66.74% Beats 74.71% Beats 87.82% AVG 78.42% NA NA	Banta 7848% Benta 82.37% Benta 82.37% AVG 80.41% NA NA Benta 94.40% AVG 94.40%	NA NA NA O Error Runtime Error Runtime Error	50(100%) 50(100%) 50(100%) 100% 50(100%) 50(100%)	Baata 84.44% Baata 78.21% Baata 33.48% AVG 65.37% Baata 50.59% Baata 50.59%	Beats 75.88% Beats 85.21% Beats 27.63% AVG 62.91% Beats 27.45% Beats 27.45% Beats 85.88% Beats 85.78% AVG 58.04%	NA NA O Error NA NA	
	Medium,	GP14 GP14 GP14 GP14 Summary Chaude Halku Chaude Halku Chaude Halku Halku Summary	50 50 50 50 50 50 50	50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 100%	Beets 73.00% Beets 88.80% Beets 80.14% AVG 66.65% Beets 90.75% Beets 90.75% Beets 91.54%	Beats 72.87% Beats 59.43% Beats 59.43% AVG 63.91% Beats 59.12% Beats 59.12% Beats 59.40% AVG 55.21%	NA NA O Error NA NA NA O Error	50(100%) 50(100%) 50(100%) 100% 11(22%) 11(22%) 50(100%)	Beets 66,74% Beets 74,71% Beets 74,71% Beets 87,82% AVG 78,42% NA Beets 85,94% AVG 85,94%	Beats 76.48% Beats 82.37% Beats 82.37% AVG 80.41% NA NA Beats 94.40%	NA NA O Error Hantense Error Hantense Error NA 2 Error	50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 100%	Basta 84.44% Basta 78.21% Basta 33.48% AVG 65.37% Basta 50.59% Basta 98.04% Basta 98.08% AVG 81.57%	Beats 75.88% Beats 85.21% Beats 27.63% AVG 62.91% Beats 27.45% Beats 85.88% Beats 60.78%	NA NA NA O Error NA NA NA O Error	
	Medium, 48.4%	GP14 GP14 GP14 GP14 Semmary Chaote Halko Chaote Halko Chaote Halko Chaote Halko Chaote Semmary Chaote Semma	50 50 50 50 50 50 50 50 50	50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%)	Besta 73.00% Besta 86.80% Besta 86.80% Besta 80.14% AVG 86.85% Besta 90.45% Besta 91.54% AVG 90.91% Besta 81.87%	Basis /2.87% Basis /0.43% Basis /0.43% Basis /0.43% Basis /0.43% Basis /0.42% Basis /0.42% Basis /0.40% AVG 56.21% Basis /0.10%	NA NA D Emor NA NA NA O Emor NA	S0(100%) S0(100%) S0(100%) 10(22%) 11(22%) S0(100%) 48% S0(100%)	Beets 66,74% Beets 74,71% Beets 74,71% AVG 78,42% NA NA Beets 85,94% AVG 85,94% Beets 87,63%	Banta 76.48% Banta 76.48% Banta 82.37% AvG 80.41% NA NA Banta 94.40% AvG 80.440% Banta 94.40% Banta 82.30%	NA NA NA O Error Hantene Error Na NA 2 Error NA	50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 100% 50(100%)	Basts 84.44% Basts 78.21% Basts 78.21% Basts 33.48% AvG 85.37% Basts 90.59% Basts 98.04% Basts 98.05% AvG 81.57% Basts 87.16%	Beats 75.88% Beats 85.21% Beats 27.65% AVG 82.91% Beats 27.45% Beats 85.88% Beats 80.78% AVG 58.04% Beats 62.3%	NA NA NA O Error NA NA O Error NA	
	Medium, 48.4% Acceptance	GP14 GP14 GP14 GP14 Summary Claude Helku Claude Helku Claude Helku Claude Helku Claude Helku Claude Helku Claude Somet	50 50 50 50 50 50 50 50 50 50	50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%)	Besta 73.00% Besta 86.80% Besta 86.80% Besta 90.45% Besta 90.45% Besta 90.45% Besta 91.54% AVG 90.91% Besta 81.87% Besta 75.32%	Bachs 72.87% Bachs 59.43% Bachs 59.43% Bachs 59.12% Bachs 59.12% Bachs 59.12% Bachs 59.12% Bachs 59.12% Bachs 59.10% Bachs 59.10%	NA NA D Emor NA NA NA O Emor NA	50(100%) 50(100%) 50(100%) 100% 11(22%) 50(100%) 48% 50(100%) 50(100%)	Beerls 66,74% Beerls 66,74% Beerls 74,71% Beerls 75,82% NA NA NA Beerls 55,94% Beerls 75,62% Beerls 71,75%	Beats 78.48% Beats 82.37% Beats 82.37% AVG 80.41% NA NA Beats 94.40% Beats 94.40% Beats 82.30% Beats 82.30%	NA NA NA O Enter Rankens Enter NA 2 Enter NA NA NA	50(100%) 50(100%) 100% 50(100%) 50(100%) 50(100%) 50(100%) 100% 50(100%) 50(100%)	Basta 84.44% Basta 78.21% Basta 78.21% Basta 50.50% Basta 50.50% Basta 50.00% Basta 50.00% Basta 50.00% Basta 51.00% Basta 61.10%	Beats 75.88% Beats 85.21% Beats 27.63% Beats 27.63% Beats 27.45% Beats 27.45% Beats 27.45% Beats 27.45% Beats 22% Beats 22% Beats 25%	NA NA NA O Error NA NA O Error NA NA	
	Medium, 48.4% Acceptance	Gir14 Gir14 Gir14 Gir14 Summery Checke Helko Checke Helko Checke Helko Checke Helko Checke Somet Checke Somet	50 50 50 50 50 50 50 50 50 50 50	S0(100%) S0(100%) S0(100%) S0(100%) S0(100%) S0(100%) S0(100%) S0(100%) S0(100%) S0(100%) 0(0%)	Beats 73.00% Beats 88.80% Beats 88.80% Beats 90.14% Beats 90.45% Beats 90.45% Beats 91.54% AVG 90.91% Beats 91.54% Beats 75.32% NA	Bach 72.87% Bach 59.43% Bach 59.43% Bach 59.43% Bach 59.12% Bach 59.12% Bach 59.12% Bach 59.12% Bach 59.10% Bach 69.11% Bach 69.11%	NA NA NA Entror NA NA O Entror NA NA NA NA	S0(100%) S0(100%) S0(100%) 10(22%) 11(22%) 50(100%) S0(100%) S0(100%) S0(100%)	Beath 66,74% Beath 76,71% Beath 74,71% Beath 87,82% AVG 78,42% NA Beath 85,94% AVG 85,94% Beath 87,83% Beath 71,75% Beath 82,47%	Beats 76.45% Beats 82.37% Beats 82.37% AVC 80.41% NA NA Beats 94.40% AVC 94.40% Beats 82.30% Beats 82.30%	NA NA NA O Estor Matema Envo NA 2 Estor NA NA NA	50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%)	Sauta 84.44% Baata 78.21% Baata 33.46% Baata 50.50% Baata 50.05% AvG 81.57% Baata 50.05% AvG 81.57% Baata 57.16% Baata 64.03% Baata 52.81%	Beats 75.85% Beats 85.21% Beats 27.85% AVG 82.91% Beats 27.45% Beats 85.85% Beats 85.85% Beats 85.95% Beats 81.05% Beats 81.05%	NA NA NA NA NA NA O Error NA NA NA	
	Medium, 48.4% Acceptance	CP14 CP14 CP14 CP14 CP14 CP145 CPaude Maileu CPaude Maileu CPaude Maileu CPaude Sonnet CPaude Sonnet CPaude Sonnet Sonnet Sonnet CPaude Opas CP14	50 50 50 50 50 50 50 50 50 50 50 50 50 5	50(100%) 50(100%) 50(100%) 500% 50(100%) 50(100%) 50(100%) 50(100%) 80(100%) 87% 50(100%) 50(100%)	Beak 23:00% Beak 85:80% Beak 85:80% Beak 80:14% Beak 90:75% Beak 90:75% Beak 90:75% Beak 90:75% Beak 91:75% Beak 91:75% Beak 91:25% Beak 90:25% Beak 90:25%	Bank 72.87% Bank 59.43% Bank 59.43% AVG 83.31% Bank 59.12% Bank 59.12% Bank 59.12% Bank 59.12% Bank 59.10% Bank 68.91% NA AVG 83.01% Bank 68.91%	NA NA NA O Emor NA NA NA NA NA NA NA	50(10%) 50(10%) 50(10%) 100% 11(22%) 50(10%) 50(10%) 50(10%) 50(10%) 50(10%) 50(10%) 50(10%)	Bauts 88,74% Bauts 74,71% Bauts 74,71% AVG 78,42% NA Bauts 85,94% Bauts 87,63% Bauts 87,63% Bauts 87,63% Bauts 87,63% Bauts 82,47% AVG 73,95% Bauts 85,94%	Besta /6.45% Besta 22.37% AVG 80.41% NA Besta 22.37% AVG 80.41% NA Besta 24.40% Besta 22.30% Besta 22.30% Besta 22.30% Besta 22.30% Besta 21.30% Besta 11.00%	NA NA NA O Error Na Z Error NA NA NA O Error NA NA	50(100%) 50(100%) 100% 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%)	Basta B4.44% Basta B4.44% Basta 73.24% AVC 85.27% Basta 80.59% Basta 80.59% AVC 81.57% Basta 80.05% Basta 81.57% Basta 82.05% Basta 82.81% AVC 82.27% Basta 82.47% Basta 82.47%	Beats 75.8% Beats 75.2% Beats 27.6% AVG 82.21% Beats 27.6% Beats 27.4% Beats 85.8% Beats 62% Beats 61.0% Beats 61.0% Beats 51.0% Beats 5.10%	NA NA Olimer NA NA Olimer NA NA NA NA NA	
	Medium, 48.4% Acceptance	G2114 G2114 G2114 G2114 Stammery Claude Haiku Olaude Haiku Olaude Haiku Olaude Somet Claude Somet Claude Somet Somet Somet Somet Somet Claude Optic Claude Optic Claude Optic	50 50 50 50 50 50 50 50 50 50 50 50 50 5	so(100%) so(100%) 100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%)	Deals 73.00% Beaks 05.30% Beaks 05.14% AVG 66.00% Beaks 00.74% AVG 90.01% Beaks 01.54% NA AVG 90.01% Beaks 01.54% NA AVG 78.00% Beaks 00.28% Beaks 00.28% Beaks 00.28%	Bearls 72.87% Bearls 59.43% Bearls 59.43% AVG 82.31% Bearls 59.12% Bearls 59.12% Bearls 59.12% Bearls 59.12% Bearls 59.12% NA AVG 82.31% Bearls 69.91% Dearls 69.94%	NA NA NA O Lator NA NA NA NA NA NA NA NA NA NA	50(10%) 50(10%) 50(10%) 10(2%) 11(2%) 50(10%) 50(10%) 50(10%) 50(10%) 50(10%) 50(10%) 50(10%)	Bearls 65, 74% Bearls 74, 71% Bearls 74, 72% AVG 26, 82% NA Bearls 85, 94% AVG 26, 94% Bearls 85, 94% Bearls 85, 94% Bearls 82, 87% Bearls 80, 57% Bearls 85, 94% Bearls 85, 75%	Beeta /6.45% Beeta 52.37% Beeta 52.37% AVC 80.41% NA Beeta 54.40% AVC 854.47% Beeta 70.05% Beeta 70.05% Beeta 71.00% Beeta 71.00% Beeta 71.00% Beeta 71.00%	NA NA NA Essor Essor NA Essor NA NA NA NA NA NA NA NA NA NA	50(150%) 50(150%) 100% 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%)	Deats D4.44% Deats D4.44% Deats 25.24% AvG d5.13% Deats 25.34% Deats 25.35% Deats 26.04% Deats 26.04% Deats 27.16% Deats 22.81% Deats 22.81% Deats 22.81% Deats 24.05% Deats 25.44%	Bearls / 5.8% Bearls / 5.2% Bearls 27.4% Bearls 27.4% Bearls 27.4% Bearls 82.7% AVG 92.0% Bearls 82.7% Bearls 82.7% Bearls 81.0% Bearls 51.0% Bearls 51.0% Bearls 5.10% Bearls 6.2%	NA NA NA O Error NA NA NA NA NA NA NA NA NA	
	Medium, 48.4% Acceptance	GY14 GY14 GY14 GY14 Sammery Claude Helau Claude Helau Claude Helau Claude Samet Claude Samet Claude Samet Claude Samet Claude Samet Claude Samet Claude Samet Claude Samet Claude Gpas Claude Gpas Claude Gpas	50 50 50 50 50 50 50 50 50 50 50 50 50 5	50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%) 60(100%) 50(100%) 50(100%) 50(100%) 50(100%) 50(100%)	Beats 73.00% Beats 05.40% AVG 66.05% Beats 05.14% AVG 90.75% Beats 07.75% Beats 07.75% Beats 07.55% Beats 07.55% Beats 07.55% Beats 07.55% Beats 04.05% Beats 04.05%	Beach 72.87% Beach 59.43% Beach 59.43% ANG 85.31% Beach 59.12% Beach 59.12% Beach 59.12% Beach 59.12% Beach 59.12% Beach 59.10% Beach 59.10% Beach 59.10% Beach 59.40% Beach 59.44%	NA NA NA D Emor NA NA NA D Emor Emor Econor Econor Econor Sola NA NA NA NA NA NA	90(100%) 90(100%) 90(100%) 10(2%) 11(22%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%)	Bearts 65, 74% Bearts 64, 71% Bearts 67, 52% AVG 26, 82% NA NA Bearts 65, 54% AVG 26, 54% Bearts 67, 65% Bearts 67, 65% Bearts 67, 65% Bearts 65, 67% Bearts 65, 97% Bearts 65, 77%	Beeda /6-45% Beeda 82-37% Beeda 82-37% NA NA NA Beeda 94-40% AVG 94-40% Beeda 70-28% Beeda 72-20% Beeda 72-20% Beeda 71-00% Beeda 71-00% Beeda 71-00%	NA NA NA O Entror Contense Entro NA NA NA NA NA NA NA NA NA NA NA NA NA	50(150%) 50(150%) 150% 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%)	Bearls D4.44% Bearls 73.21% Bearls 73.21% Bearls 73.24% Bearls D5.0% Bearls D5.0% Bearls D5.0% Bearls D5.0% Bearls D5.0% Bearls A0.0% Bearls A0.0% Bearls A0.0% Bearls A0.0% Bearls A0.0% Bearls A0.0% Bearls A0.0%	Beeds 75.87% Beeds 55.21% Beeds 25.21% Beeds 25.27% Beeds 55.87% Beeds 55.87% Beeds 55.87% Beeds 51.75% Beeds 51.75% Beeds 51.75% Beeds 51.75% Beeds 51.75% Beeds 50.77%	NA NA NA Error NA NA NA NA NA NA NA NA NA NA NA NA	
	Medium, 48.4% Acceptance	G114 G114 G114 Sammery Claudo Helau Claudo Helau Claudo Helau Claudo Helau Helau Sameri Claudo Sameri Claudo Sameri Claudo Sameri Claudo Sameri Claudo Opas Claudo Opas Claudo Opas Claudo Opas Claudo Opas	50 50 50 50 50 50 50 50 50 50 50 50 50 5	so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%)	Beats 73.00% Beats 65.00% Beats 65.14% AVG 92.05% Beats 90.75% Beats 90.75% Beats 90.75% Beats 91.54% AVG 90.91% NA AVG 90.91% Beats 90.95% Beats 90.95% Beats 94.01% AVG 71202% Beats 94.01%	Beenty 72.87% Beenty 50.43% ANC 92.34% Beenty 50.43% Beenty 50.42% Beenty 50.12% Beenty 50.12% Beenty 50.12% Beenty 50.40% ANC 92.11% Beenty 50.40% Beenty 50.40% Beenty 50.40% Beenty 50.40% Beenty 50.40% Beenty 50.40%	NA NA NA Emer NA NA NA NA NA NA NA NA NA NA NA NA	90(100%) 90(10%) 900% 100% 11(22%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%)	Beerls 65, 74% Beerls 65, 74% Beerls 67, 52% AVG 25, 82% NA Beerls 57, 55% Beerls 57, 55% Beerls 62, 67% Beerls 62, 67% Beerls 62, 67% Beerls 62, 67% Beerls 55, 75%	Bents 78 44% Bents 72 44% Bents 82 27% AVC 82 45% NA NA Bents 92 44% AVC 104 40% Bents 82 20% Bents 72 20% Bents 71 20% Bents 71 20% Bents 71 20% Bents 71 20% Bents 71 20% Bents 72 27% AVC 72 7% Bents 72 27%	NA NA NA Essor Essor NA Essor NA NA NA NA NA NA NA NA NA NA	90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%)	Dents D4.44% Dents D4.44% Dents 23.24% AVG 85.37% Dents 50.55% Dents 50.55% Dents 50.45% Dents 50.45% Dents 50.45% Dents 20.45% Dents 20.45% Dents 25.47% Dents 25.45% Dents 25.45%	Beats /5.8% Beats /5.2% Beats 27.6% Beats 27.6% Beats 27.4% Beats 27.4% Beats 62.7% Beats 62.7% Beats 62.7% Beats 62.7% Beats 5.10% Beats 5.10% Beats 41.5% Beats 41.5% Beats 41.5%	NA NA NA NA NA NA NA NA O Error NA NA NA NA NA NA	
	Medium, 48.4% Acceptance	Gr14 Gr14 Gr14 Gr14 Surmary Daulo Halau Daulo Halau Daulo Halau Daulo Halau Halau Surmary Cloude Surmal Daulo Surmal Daulo Surmal Surmary Cloude Surmary Cloude Surmary Cloude Surmary Cloude Surmary Cloude Surmary Cloude Surmary Cloude Surmary Cloude Surmary Cloude Surmary Cloude Surmary Hamman 1 Human 1	50 50 50 50 50 50 50 50 50 50 50 50 50 5	99(10%) 99(10%) 99(10%) 10% 99(10%) 99(10%) 99(10%) 99(10%) 99(10%) 99(10%) 99(10%) 99(10%) 99(10%) 99(10%) 99(10%)	Beats 73.00% Beats 05.40% AVG 06.80% Beats 05.14% AVG 06.80% Beats 05.45% Beats 05.45% Beats 05.45% Beats 05.45% NA AVG 75.00% Beats 05.05% Beats 04.05% Beats 05.05% Beats 50.15% Beats 50.15%	Benth 72.87% Benth 59.43% Benth 59.43% ANG 83.31% Benth 50.42% Benth 50.12% Benth 50.12% Benth 50.12% Benth 50.12% Benth 50.10% Real 50.10% Benth 50.10% Benth 50.40% Benth 50.40% Benth 50.40% Benth 50.40% Benth 50.40% Benth 50.40%	NA NA NA D Emor NA NA NA D Emor Lenor NA NA NA NA NA NA NA NA	90(10%) 91(10%) 91(10%) 100% 11(12%) 91(10%) 90(10%) 9	Bearls 68, 74% Bearls 68, 74% Bearls 87, 82% AVG 28, 82% NA NA Bearls 65, 94% AVG 85, 94% Bearls 67, 85% Bearls 67, 85% Bearls 67, 85% Bearls 65, 94% Bearls 65, 94%	Data (76.47); Bank (72.27); AvG (72.47); NA Bank (72.7); AvG (72.47); Bank (74.47); AvG (78.27); Bank (72.97); Bank (72.97); Bank (72.97); Bank (72.7); Bank (72.	NA NA NA O Entror Contense Entro NA NA NA NA NA NA NA NA NA NA NA NA NA	50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%) 50(150%)	Bearls D4 A475 Bearls 78 2175 Bearls 28 2475 Bearls 28 2475 Bearls D8 D475 Bearls D1 275 Bearls D5 J475 Bearls D5 J475	Beeds 75.85% Beeds 55.21% Beeds 27.65% AVG 82.23% Beeds 85.85% Beeds 85.85% AVG 36.04% Beeds 85.85% Beeds 81.05% Beeds 91.75% AVG 35.85% Beeds 91.75% AVG 35.85% Beeds 91.75% AVG 35.85% Beeds 91.75% Beeds 41.57%	NA NA NA NA NA NA NA NA NA NA NA NA NA N	
	Medium, 48.4% Acceptance	G114 G114 G114 Sammery Claudo Helau Claudo Helau Claudo Helau Claudo Helau Helau Sameri Claudo Sameri Claudo Sameri Claudo Sameri Claudo Sameri Claudo Opas Claudo Opas Claudo Opas Claudo Opas Claudo Opas	50 50 50 50 50 50 50 50 50 50 50 50 50 5	so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%) so(100%)	Beats 73.00% Beats 65.00% Beats 65.14% AVG 92.05% Beats 90.75% Beats 90.75% Beats 90.45% AVG 90.97% Beats 75.32% NA AVG 90.97% Beats 90.95% Beats 90.95% Beats 94.07% AVG 71202% Beats 94.07%	Beenty 72.87% Beenty 50.43% ANC 92.34% Beenty 50.43% Beenty 50.42% Beenty 50.12% Beenty 50.12% Beenty 50.12% Beenty 50.40% ANC 92.11% Beenty 50.40% Beenty 50.40% Beenty 50.40% Beenty 50.40% Beenty 50.40% Beenty 50.40%	NA NA NA Emer NA NA NA NA NA NA NA NA NA NA NA NA	90(100%) 90(10%) 900% 100% 11(22%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%) 90(10%)	Beerls 65, 74% Beerls 65, 74% Beerls 67, 52% AVG 25, 82% NA Beerls 57, 55% Beerls 57, 55% Beerls 62, 67% Beerls 62, 67% Beerls 62, 67% Beerls 62, 67% Beerls 55, 75%	Bents 78 44% Bents 72 44% Bents 82 27% AVC 82 45% NA NA Bents 92 44% AVC 104 40% Bents 82 20% Bents 72 20% Bents 71 20% Bents 71 20% Bents 71 20% Bents 71 20% Bents 71 20% Bents 72 27% AVC 72 7% Bents 72 27%	NA NA NA O Entror Contense Entro NA NA NA NA NA NA NA NA NA NA NA NA NA	90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%) 90(100%)	Dents D4.44% Dents D4.44% Dents 23.24% AVG 85.37% Dents 50.55% Dents 50.55% Dents 50.45% Dents 50.45% Dents 50.45% Dents 20.45% Dents 20.45% Dents 25.47% Dents 25.45% Dents 25.45%	Beats /5.8% Beats /5.2% Beats 27.6% Beats 27.6% Beats 27.4% Beats 27.4% Beats 62.7% Beats 62.7% Beats 62.7% Beats 62.7% Beats 5.10% Beats 5.10% Beats 41.5% Beats 41.5% Beats 41.5%	NA NA NA NA NA NA NA NA O Error NA NA NA NA NA NA	

Figure 5: Appendix for All Sorting Questions Collected



Figure 6: Appendix for All Sorting Questions Collected

```
ChatGPT
```

To solve the problem of finding the longest palindromic substring in a given string, we can utilize several approaches, each with its own trade-offs in terms of time and space complexity. Here are three common methods:

 Dynamic Programming (DP): In this approach, we use a table to determine whether substrings are palindromic and then build from there to larger substrings. It requires '0(a*2)' time and '0(a*2)' space.

 Expand Around Center: This is a more space-officient method where we consider each character (and each pair of consecutive characters) as potential centers of palindromes and expand outwards. It also requires "o(n*2)" time but only "o(3)" extra space.

 Manacher's Algorithm: This algorithm finds the langest palindromic substring in linear time, ¹0(n)⁺, but it is more complex and less commonly used in practice unless optimal performance is necessary.

Given the constraint that the string length can be up to 1000, the Expand Around Center approach offers a good balance between simplicity and efficiency, thus, 111 implement that method.

Expand Around Center Approach:

The core idea of this method is that a palindrome mirrors around its center. Therefore, a palindrome can be expanded from its center, and there are "2n-1" such centers (considering every individual character and the gaps between them as potential centers).

Here's how the method works:

- · Iterate through each possible center of the palindrome.
- Expand around each center while the characters on both sides match.
- Keep track of the longest palindrome found during these expansions.

Let's implement this in Python:

pythan Coxy cod	
def longestPalindrowe(self, s: str) -> str:	
1f len(s) == 0:	
return "	
start, end = 0, 0	
<pre>def expand_around_center(left: int, right: int) -> None:</pre>	
nonlocal start, end	
while left >= 0 and right < len(s) and s[left] == s[right]:	
left 1	
right += 1	
# After expanding, left and right are one step further out from valid	
if (right - left - 1) > (end - start + 1):	
start = left + 1	
and = right - 1	
uni - rigin - l	
<pre>for 1 in range(len(s)):</pre>	
# Odd length palindromes: Expand from one character	
expand_arcund_center(1, 1)	
# Even length palindromes: Expand from between characters	
if i + i < lem(s):	
expand_around_center(1, 1 + 1)	
return s[startrend+1]	
Analysis:	
 Time Complexity: Each expansion in the worst case can take 10(n) 1 time and since we perform 	
expansion around '2n-1' centers, the total time complexity is '0(n*2)'.	
 Space Complexity: We use only a few variables to keep track of indices, so the space complexity is 	
'0(1)'.	
This solution efficiently balances time complexity and space usage, while being relatively	
straightforward to implement. It also works effectively within the given constraints.	
ውሮ ኮም	
•	
Message ChatGPT.	
ChatGPT can make mistalies. Consider checking important information.	

Figure 7: Appendix for Example ChatGPT Answer

To solve this problem, we need to find all the possible palindromic substrings in the given. string s and return the longest one. A palindromic string is a string that reads the same backward as forward, like "racecar" or "level". Here's a Python solution using the expanding around the center approache C Copy code Explanation: 1. The expand_around_center function takes two indices left and right as input and expands outward from the center until it finds the maximum palindrome around that center. 2. In the longestPalindrone function, we first handle the edge case where the input string s is empty by returning an empty string. 3. We initialize an empty string longest_palindrone to store the longest palindromic substring found so far. 4. We iterate through each character in the string s-using a loop. 5. For each character, we consider two cases: Odd length palindromes: We call expand_around_center with i as both left and right indices, effectively treating the current character as the center of the palindrome. Even length palindromes: We call expand_around_center_with 1 and 1 + 1 the next character as the center of the palindrome. 6. After expanding around each center, we update longest_palindrone with the longer palindromic substring found.

 Finally, we return longest_palindrone, which contains the longest palindromic substring in s.

Assumptions:

- · The input string a consists of only digits and English letters, as per the constraints.
- The length of the input string is its within the specified range of 1 <= s. length <= 1993

Time Complexity:

The time complexity of this solution is O(n⁻²), where n is the length of the input string s. This is because in the worst case, we need to expand around each character and each pair of characters, which takes O(n) time for each expansion. Since we have stimost 2n centers (n characters and n - 1 pairs of characters), the overall time complexity is O(n⁺2).

Space Complexity:

⋇

The space complexity of this solution is U(1), as we are not using any additional data structures that scale with the input size. We are only using a constant amount of estra space to store temporary variables.

 $\underline{\mathbb{A}}_{\mathbb{A}}$. Clearle does not have the shifty to run the code it generates yet

ñ Capy O Retry id. 4

Figure 8: Appendix for Example Claude Answer