B.3.3 USER UTTERANCE GENERATION

Once conversation plans are constructed, user turns are synthesized directly from them. Each sub-plan within a conversation plan consists of M bullet-points, which are partitioned into K contiguous batches of equal size. Partitioning is performed sequentially, such that each batch corresponds to a consecutive segment of the sub-plan. Partitioning is necessary because conditioning the LLM on an entire sub-plan at once tends to yield repetitive or low-quality questions; batching mitigates this by narrowing the focus of generation. For each batch, the LLM produces I user questions (line 6 of Algorithm 2 in Appendix B.3.5) using the prompt presented in Listing 32. The model is conditioned on the conversation seed, the current batch specification, preceding batches within the same sub-plan, and contextual information from earlier sub-plans. This setup ensures that generated questions remain grounded in prior context, yielding conversations that are coherent and continuous over extended spans.

The values of K and I vary depending on the domain and the target conversation length, in order to adhere to the overall length budget. We specify the values for K and I manually. The specific configurations of K and I across domains and conversation sizes are reported in Table 6. This provides fine-grained control over the density of user interactions and helps prevent both under-generation and excessive redundancy. Additonally, to better capture domain-specific conversational patterns, we incorporate domain-specific features during question generation:

- **Programming:** To reflect realistic developer–assistant interactions, we incorporate questions that involve sharing code snippets. These include (i) buggy code requiring debugging assistance, (ii) correct code seeking optimization, and (iii) natural language descriptions of desired functionality for which code is requested. We use the prompt shown in Listing 33 to generate questions specific to the programming domain.
- **Mathematics.** To capture authentic problem-solving dynamics, we incorporate questions that involve sharing mathematical work, requesting corrections, asking for the next logical step in a solution, or introducing problems to be solved. We use the prompt shown in Listing 34 to generate questions specific to the mathematics domain.

To reduce computational cost while maintaining generation quality, question generation is performed using the open-source LLaMA-3.3 70B model (AI, 2024), which produces high-quality questions.

B.3.4 ASSISTANT UTTERANCE GENERATION

After generating user-side questions, assistant-side responses are generated in an iterative, role-playing framework where one LLM assumes the *assistant role* and another assumes the *user role*. For each sub-plan, the assistant LLM is conditioned on the seed as explained in Section 2.2.1, prior sub-plans of the conversation plan, a summary of the most recent M dialogue turns, and a compressed summary of older turns (generated using the prompt shown in Listing 37). For 10M-token conversations, additional summaries of prior plans are also provided.

The response generation process unfolds as an iterative interaction between the assistant and user roles. First, the assistant LLM produces an answer to the user's most recent question (line 9). This output is then analyzed by a *question-detection module*, which determines whether the assistant's response contains a counter-question directed at the user (line 11), using the prompt shown in Listing 35 that takes the assistant response as input and outputs `yes` if a question is present and `no` otherwise. If such a counter-question is detected, the response—together with the current and previous sub-plans, relevant past context, and conversation summaries—is passed to the user LLM, which generates a realistic reply that reflects the storyline and contextual details using the prompt shown in Listing 38 (line 14). This new user reply is subsequently passed back to the assistant LLM, continuing the conversation. This loop repeats until no further assistant questions are detected or the predefined threshold δ_1 (which is set to two) is reached, preventing infinite cycles. For δ_1 we tested values 2, 3 and 5 which we selected 2 as it produces more realistic dialogues.

Beyond direct question–answer exchanges, a *follow-up detection module* (line 21) evaluates whether, in a realistic setting, the user would naturally ask a clarifying or elaborative follow-up. The need for a follow-up is determined using the prompt shown in Listing 36, which takes as input the seed, dialogue history, and the assistant's most recent response, and outputs `yes` or `no`. This