Assignment-3 Gen AI

Ques-Create a multi step lang chain pipeline that takes user input, reformulates it using a prompt template, and generates a summary define.

Step 1: Import Required Libraries

We first import LangChain components needed for LLMs, prompt templates, and chains.

```
from langchain.llms import OpenAI
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain, SequentialChain
```

Step 2: Initialize the Language Model (LLM)

Create an LLM instance that will process text.

```
llm = OpenAI(temperature=0.3)
```

Explanation:

- `temperature=0.3` → gives more focused and less random responses.

Step 3: Create Prompt Template for Reformulation

This step rewrites or improves the user input.

```
reformulate_prompt = PromptTemplate(
    input_variables=["user_input"],
    template="Rewrite the following text in a clear and well-
structured way:\n\n{user_input}"
)
```

Step 4: Create Reformulation Chain

This chain uses the LLM and reformulation prompt.

```
reformulation_chain = LLMChain(
    llm=llm,
    prompt=reformulate_prompt,
    output_key="reformulated_text"
)
```

Output:
➡ Cleaned / rewritten version of the user input

Step 5: Create Prompt Template for Summarization

Now we define how the summary should be generated.

```
summary_prompt = PromptTemplate(
    input_variables=["reformulated_text"],
    template="Summarize the following text in 3-4
lines:\n\n{reformulated_text}"
)
```

Step 6: Create Summary Chain

```
summary_chain = LLMChain(
    llm=llm,
    prompt=summary_prompt,
    output_key="summary"
)
```

Output:
➡ Short, meaningful summary

Step 7: Combine Chains into a Sequential Pipeline

We now connect all steps using `SequentialChain`.

```
pipeline = SequentialChain(
    chains=[reformulation_chain, summary_chain],
    input_variables=["user_input"],
    output_variables=["reformulated_text", "summary"]
)
```

Step 8: Execute the Pipeline with User Input

```
result = pipeline.run(
    user_input="Machine learning is a field of AI that allows
systems to learn from data and improve performance without being
explicitly programmed."
)

print("Reformulated Text:\n", result["reformulated_text"])
print("\nSummary:\n", result["summary"])
```

Final Workflow Diagram (Conceptual)

```
User Input
   ↓
```

```
Prompt Template (Reformulation)
   ↓
LLMChain (Rewrite Text)
   ↓
Prompt Template (Summarization)
   ↓
LLMChain (Generate Summary)
   ↓
Final Output
```

 Conclusion

This multi-step LangChain pipeline demonstrates how **SequentialChain** can be used to:

- Process user input in stages
- Apply different prompt templates
- Produce structured and meaningful outputs

Such pipelines are useful in **chatbots, document processing, content generation, and AI assistants**.