Names: Ivana Miovcic
Mishelle Bitman
Daniel Persaud

For this question, we chose ChatGPT as our LLM since it is a very large and free LLM to use. We decided to try Q2 from assignment 1 and split the question up into 3 parts: 2. a), 2. b) and 2. c). We wanted to present the information to the LLM in a way that felt the most similar to how a person would read and do the question. To accomplish that we copied and pasted each part of question 2 into ChatGPT as prompts.

ChatGPT did question 2 a) with no mistakes. It created all the predicates needed for our knowledge base and stuck to the guidelines for the question. We were surprised to see how quick and efficient the process was, although we noticed that ChatGPT stuck to the minimum requirements. For example, question 2 a) asks that you create at least 2 people with bank accounts at cibc and bmo, and ChatGPT created exactly 2 people for its solution. We double checked the output and there were no mistakes.

```
% Define accounts for two people
hasAccount(john, cibc, 1000).
hasAccount(john, bmo, 800).
hasAccount(mary, cibc, 1200).
hasAccount(mary, bmo, 1500).

% Total deposits by each person at each bank
totalDeposits(john, cibc, 500).
totalDeposits(john, bmo, 300).
totalDeposits(mary, cibc, 700).
totalDeposits(mary, bmo, 400).
```

For part 2 b) ChatGPT used some symbols that we were not allowed to use in the assignment. We had to ask it to redo the question without using the “->” symbol. ChatGPT also tried using the findall predicate in prolog, so we had to ask it to try recreating the endOfMonthBalance predicate without using findall. When we asked it to redo the question it used the “\+” symbol which we did not cover in class, but a quick google search told us it was equivalent to the not operator. Once again, chatGPT was able to answer the question relatively correctly but with a few minor issues that were fixable.

Before:

```
% Calculate accrued interest for each person's account at the specified bank
accruedInterest(Name, Bank, Interest) :-
    subtotal(Name, Bank, Subtotal),
    interestLevel(Bank, MinLevel),
    (   Subtotal >= MinLevel
    ->  monthlyRate(Bank, Rate),
        Interest is Subtotal * Rate
    ;   Interest is 0).
```

```
% Calculate end-of-month balance for each person's accounts across all banks
endOfMonthBalance(Name, TotalBalance) :-
    findall(Balance, endOfMonthBalance(Name, Bank, Balance), Balances),
    sum_list(Balances, TotalBalance).
```

After:

```
% Calculate accrued interest for each person's account at the specified bank
accruedInterest(Name, Bank, Interest) :-
    subtotal(Name, Bank, Subtotal),
    interestLevel(Bank, MinLevel),
    MinLevel =< Subtotal,  % Check if subtotal meets minimum level
    monthlyRate(Bank, Rate),
    Interest is Subtotal * Rate.
```

```
% Helper predicate to accumulate total balance
totalEndOfMonthBalance(Name, TotalBalance, BalanceAccum) :-
    endOfMonthBalance(Name, Bank, Balance),
    NewAccum is BalanceAccum + Balance,
    totalEndOfMonthBalance(Name, TotalBalance, NewAccum).

totalEndOfMonthBalance(Name, TotalBalance, 0) :-
    \+ endOfMonthBalance(Name, _, _).  % Base case: no balances found.

% Calculate end-of-month balance for each person's accounts across all banks
endOfMonthBalance(Name, TotalBalance) :-
    totalEndOfMonthBalance(Name, TotalBalance, 0).
```

It even gave us queries we could try to test the question which we did not ask it to do. We tried all the queries and they returned the correct answers.

> You can query your Prolog knowledge base with commands such as:
>
> - `subtotal(john, cibc, X).`
> - `accruedInterest(mary, bmo, Y).`
> - `accruedPenalty(john, cibc, Z).`
> - `endOfMonthBalance(mary, TotalBalance).`

Lastly, we tried inputting 2 c) as a prompt. It creates 2 queries for each predicate, one to test Mary's account and the other to check John's account. The queries all worked when we tested them.

```
% Check the subtotal for Mary's account at BMO
?- subtotal(mary, bmo, Y).
% This checks what the subtotal amount is for Mary's account at BMO.

% Check the accrued interest for John's account at CIBC
?- accruedInterest(john, cibc, I).
% This checks how much interest John earns on his CIBC account.
```

We were surprised to find that chatGPT did a very good job solving this question although there were some parts that we had to ask it to redo for not using the correct syntax, which slowed down answering the question. After testing the predicates it created with our own queries, as well as the queries from part 2 c), we found that there were no errors in the output code. ChatGPT managed to answer the question correctly and relatively efficiently.