Relational Algebra Autograder Instructions

- Please use the provided LaTeXtemplate for your submissions and do not edit this file outside the answer sections.
- Please only use the defined Latex commands defined in Table 1
- Please follow attribute prefixing rules, when dealing with self joins or theta joins. Please check the Attribute Prefixing section for more details.
- Each RA statement should be on a separate line and surrounded by $ signs. The $ signs should only be at the start and end of the RA expression and not inside it.
- If using multiple intermediate steps and creating temporary relations, you need to mention the relation that contains the final answer as the last expression for the answer.
- Do not surround relation names with curly braces {}
- Only use the $\leftarrow$ for assignment expressions. Do not use "=", ":" or anything else.
- Use \geq, \leq, and \neq for greater than equal, less than equal and not equal conditions

Name	**Latex Sequence**	**Symbol**
Assignment	\leftarrow	$\leftarrow$
Natural Join	\bowtie	$\bowtie$
Theta Join	\bowtie_{cond}	$\bowtie_{cond}$
Left Outer Join	\leftouterjoin	$⟕$
Right Outer Join	\rightouterjoin	$⟖$
Full Outer Join	\fullouterjoin	$⟗$
Cross Join	\times	$\times$
Projection	\pi	π
Selection	\sigma_{cond}	σ_{cond}
Rename	\rho	ρ
Union	\cup	$\cup$
Intersection	\cap	$\cap$
Difference	-	$-$
AND	\land	$\land$
OR	\lor	$\lor$

Table 1: Relational Algebra Operators Latex Sequence and symbols

Basic Usage

This is a guide on how to use each RA operator. The examples given are based on the airbnb database schema, which is:

Table	Attributes
host	host_id host_name
area	area_id area
neighbourhood	neighbourhood_id name area_id
room	room_type_id room_type
place	id name host_id neighbourhood_id latitude longitude room_type_id price minimum_nights number_of_reviews reviews_per_month availability_365

Table 2: AirBnB Database Schema

Selection (σ): \sigma_{condition} (input_relation)

For example, to select places in the range of $100 and $200, we can write:

```
\sigma_{\text{p.price} \geq 100 \land \text{p.price} \leq 200} (\text{place p})
```

$$\sigma_{\text{p.price} \geq 100 \land \text{p.price} \leq 200}(\text{place p})$$

Note that, enclosing relation names, aliases and attributes in the \text{} block is **optional**, but that results in better formatted and more human readable RA queries. If this is not enclosed in a \text{} block, it is rendered in italics and without spacing
In another example, for getting the details of "Brooklyn Heights" neighbourhood, we can write:

```
\sigma_{\text{n.name} = \text{'Brooklyn Heights'}} (\text{neighbourhood n})
```

Note that string literal should be encosed within single or double qoutes

$$\sigma_{\text{n.name='Brooklyn Heights'}}(\text{neighbourhood n})$$

Projection (π): \pi_{attribute list} (input_relation)

The attribute list is a comma-seprated list of expressions that specifies the output attributes.
For example, to get the name and price of places that have been reviewed at least 50 times, we can write:

```
\pi_{\text{p.name, p.price}} (\sigma_{\text{p.number_of_reviews} \geq 50 } (\text{place p}))
```

$$\pi_{\text{p.name, p.price}}(\sigma_{\text{p.number\_of\_reviews}\geq 50}(\text{place p}))$$

Remember to only project those attributes that do exist in the given input relation.

Natural Joins ($\bowtie, \rtimes, \ltimes, \rtimes\!\ltimes$): relation1 \[bowtie|leftouterjoin|rightouterjoin|fullouterjoin] relation2

Natural joins can only be performed between relations having similarly named attribute\s. For example, to get the host information of each place we can perform a natural join between the host and place relations, as they share an identically named attribute, host_id.

```
\pi_{\text{p.host_id, p.name, h.host_name}} (\text{place p} \bowtie \text{host h})
```

$$\pi_{\text{p.host\_id, p.name, h.host\_name}}(\text{place p} \bowtie \text{host h})$$

Natural join will automatically equate all pairs of identically named attributes from its inputs (in this case, host_id), and the output will contain only one attribute per pair, which can be referenced with relation1. So, the resulting relation will only have one column for host_id, which can be only accessed using p.host_id and not h.host_id.

Theta Joins ($\bowtie_\theta, \rtimes_\theta, \ltimes_\theta, \rtimes\!\ltimes_\theta$): relation1 \[bowtie|leftouterjoin|rightouterjoin|fullouterjoin]_{condition} relation2

Theta joins allow us to specify an explicit condition for joining relations, instead of relying on identically named attributes like in a natural join. For example, to get the host information of each place, we can perform a theta join between the host and place relations by explicitly specifying the equality condition.

```
\text{place p} \bowtie_{\text{p.host_id = h.host_id}} \text{host h}
```

$$\text{place p} \bowtie_{\text{p.host\_id = h.host\_id}} \text{host h}$$

Note that in a theta join, if there are attributes across the relations sharing the same name, they will be prefixed with the relation name or the alias followed by _ Hence the intermediate relation storing the result of the theta join will contain both copies of the common attributes. Please refer to the attribute prefixing section for more details.

Cross product ($\times$): relation1 \times relation2

The cross product (Cartesian Product) returns all possible combinations of tuples from the two relations involved. For example, to get the cross product of the neighbourhood and area relations, we can write:

$$\text{neighbourhood} \times \text{area}$$

When using cross joins with an assignment operator, there will be attribute prefixing if similarly named attributes exist. Please refer to the Attribute Prefixing section for more details.

Set Operations ($\cup, \cap, -$): relation1 \[cup|cap|-] relation2

When applying the set operations, you need to have an equal number of attributes across the relations you want to apply the set operation.

Assignment ($\leftarrow$): variable_name \leftarrow RA Expresion

In order to simplify a complex RA expression, you can break it down into several intermediate RA expressions. And that can be made possible using the assignment operator, where you can store the result of any RA expression in an intermediate relation, which can be referenced later. For example, to get place id and name of places hosted by "Jake", we can write:

$$a \leftarrow \text{place p} \bowtie \text{host h}$$

Now, using the relation a, we can filter for places hosted by "Jake" and retrieve their id and names:

$$\pi_{\text{a.id, a.name}}(\sigma_{\text{a.host\_name} = \text{'Jake'}}(\text{a}))$$

Attribute Prefixing

Whenever using an assignment operation, where the RA expression contains a Theta Join, Cross Joins or self-joins we perform attribute prefixing. This means that if the relations on which the join is performed have attributes sharing similar names, then those attribute names are prefixed with the relation name they come from. For example lets get the ids and names of the places in the neighbourhood of "Harlem" using a cross join.

$$\begin{gathered} a \leftarrow \text{place p} \times \text{neighbourhood n} \\ b \leftarrow \sigma_{\text{a.n\_neighbourhood\_id = a.p\_neighbourhood\_id}}(a) \\ c \leftarrow \sigma_{\text{b.n\_name = 'Harlem'}}(b) \\ d \leftarrow \pi_{\text{c.id, c.p\_name}}(c) \\ d \end{gathered}$$

Lets break down this example, the first RA expression takes a cross product between the place and neighbourhood relations. Since, place and neighbourhood share two similarly named attributes (name and neighbourhood_id), they are renamed to p_name, n_name, p_neighbourhood_id and n_neighbourhood_id. Where the prefix of $p\_$ suggests the attribute is from the place relation and the prefix of $n\_$ suggests the attributes are from the neighbourhood relation. You can observe in the rest of the RA expressions, how I have used these prefixed attributes.

Lets take a look at another example, this time involving a self join. In a self join, every attribute will be prefixed as the join happens between the same relation. Lets get pairs of places that have the same price.

$$a \leftarrow \text{place p1} \bowtie_{\text{p1.price = p2.price} \wedge \text{p1.id} \neq \text{p2.id}} \text{place p2}$$
$$b \leftarrow \pi_{\text{a.p1\_id, a.p1\_name, a.p2\_id, a.p2\_name, a.p2\_price}}(a)$$
$$b$$

Also, do notice that as the last RA expression, we just mention the intermediate relation that stores our final answer.

Common Errors

Here is a list of common errors, students might encounter while dealing with the autograder.

- **Not mentioning the relation that contains your final result at the end.** Always ensure that the last line of your RA expression contains the relation holding your final result.

  Incorrect:
  $a \leftarrow ...$
  $b \leftarrow ...$
  (Missing final relation reference)

  Correct:
  $a \leftarrow ...$
  $b \leftarrow ...$
  b

- **Having more than one join operator in a single RA expression.** The autograder may not support multiple joins in a single RA step, so break them into multiple intermediate steps.

  Incorrect:
  $a \leftarrow c \bowtie d \bowtie e$

  Correct:
  $a \leftarrow c \bowtie d$
  $b \leftarrow a \bowtie e$

- **Using relations not present in the RA expression inside projection and selection operators.** You can only reference attributes from the last assigned relation in projection or selection.

Incorrect:

$$a \leftarrow \text{place p} \bowtie \text{host h}$$
$$\pi_{\text{p.name}}(a)$$

Correct:

$$a \leftarrow \text{place p} \bowtie \text{host h}$$
$$\pi_{\text{a.name}}(a)$$

- **Having intermediate $ signs in RA expressions.** Only use $ at the beginning and end of the entire RA expression, not in the middle of individual steps.