Answer the questions in the spaces provided on the question sheets. You should have plenty of room for the answers, but if you run out of room because you need to make a correction, you can use the back of the page (and indicate that you are doing so).

The first page of the quiz is a reference for list and table operations. Unless otherwise noted in a question, feel free to use any of the operations defined in the reference. Please do not use operations that are not in the reference.

Answer key

Anonymous grading ID: ____________________________

Name: ____________________________________________

(Please include your name if and only if you don’t remember your anonymous ID)

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understanding operations</td>
<td>20</td>
</tr>
<tr>
<td>The order-by-total function</td>
<td>20</td>
</tr>
<tr>
<td>The good-list function</td>
<td>25</td>
</tr>
<tr>
<td>Total:</td>
<td>65</td>
</tr>
</tbody>
</table>
The first two questions use the following table:

\[
\text{orders = table: product :: String, unit-price :: Number, quantity :: Number, discount-code :: String}
\begin{align*}
\text{row: "Warm hat", 10, 17, ""} \\
\text{row: "Winter coat", 50, 2, ""} \\
\text{row: "Scarf", 12, 10, "CHEAPSCARF"}
\end{align*}
\]

Understanding operations  (20 points)
Examine each of the expressions below. Next to each expression, write the output it produces when it is entered in Pyret’s “Interactions” window. None of the programs produce errors.

The orders table, above, has been defined in Pyret’s “Definitions” window.

If an expression returns a list, write it out: for instance, an answer might be

```
[1, 2, 3]
```

(a) 5 points Expression 1

\[
\text{orders.row-n(0)["product"]}
\]

(a) “Warm hat”

(b) 5 points Expression 2

\[
\text{filter-by(orders, lam(r):}
\begin{align*}
&\text{(r["discount-code"] == ") and
&\text{(r["quantity"] < 15)
&end).row-n(0)["unit-price"]}
\end{align*}
\]

(b) 50

(c) 5 points Expression 3

\[
\text{filter-by(orders, lam(r):}
\begin{align*}
&\text{(r["quantity"] > 5)
&end).get-column("quantity")}
\end{align*}
\]

(c) [17, 10]

(d) 5 points Expression 4

\[
\text{L.map(lam(x): x + 2 end, [list: 1, 2, 3])}
\]

(d) [3, 4, 5]
The order-by-total function (20 points)

You've been asked to write a function called order-by-total, which takes a table with the same structure as the orders table (as defined above). It should add a column to the table called total-price, which contains the unit-price and quantity columns multiplied together. It should return the resulting table, sorted by the values in the new column in descending order. It should pass the following test:

orders-answer = table: product :: String,
  unit-price :: Number,
  quantity :: Number,
  discount-code :: String,
  total-price :: Number
row: "Warm hat", 10, 17, "", 170
row: "Winter coat", 50, 2, "", 100
row: "Scarf", 12, 10, "CHEAPSCARF", 120
end
order-by-total(orders) is orders-answer

Your function doesn’t need to include a docstring or tests, but it should be correctly annotated with types.

fun order-by-total(t :: Table) -> Table:
  order-by(
    build-column(t, "total-price",
      lam(r):
        r["unit-price"] * r["quantity"]
      end),
    "total-price",
    false)
end

(note: students may use a helper function instead of the lambda)
The **good-list function**  (25 points)

You are working for a company with a bug in their database software: it will crash if any of the numbers 10, 25, or 2019 appear in an input. As such, you’ve been asked to write a function to ensure that lists do not contain any of these troublesome numbers.

(a) **10 points** bad-number

Write a function `bad-number` that takes a Number and returns a Boolean. It should return `true` is the number is 10, 25, or 2019 and `false` otherwise. Hint: this function should not require any list or table functions.

```plaintext
fun bad-number(x :: Number) -> Boolean:
  (x == 10) or (x == 25) or (x == 2019)
end
```
(b) The good-list function

Fill in the placeholders in the following definition of the good-list function, which checks to see if any of the bad numbers are present in the list. The function should return false if a bad number is present, and true otherwise. The function should call bad-number in order to determine whether a number is bad.

fun good-list(lst :: List<Number>) -> Boolean:
  doc: "determines whether lst is safe for the database"
  cases (List) lst:
    | empty => Empty case
    | link(fst, rst) => Link case
  end

where:
  good-list([list: 1, 2019, 2]) is false
  good-list([list: 1, 2, 3]) is true
end

i. 5 points Empty case

true

ii. 10 points Link case

not(bad-number(fst)) and good-list(rst)