Final Project Demo CS 4110

1)
The first demo displays the language and some of its capabilities. After setting up your environment through the readme, run `trauma tests/differential_dataflow/demo_1d.tr`.

The comments in the code will explain well what is going on in the code but basically you will see that we first start by defining collections, adding those collections to traces as differences of versions, first printing out the versions (sum of differences), applying a differential operator (distinct) on that trace, and then printing out the versions of the output trace. Distinct is an operator that makes a value 1 if its greater than 1 and 0 otherwise.

If you look at the output, you will see an INPUT VERSIONS section which would show you how the input evolves and OUTPUT VERSIONS section which would show you the corresponding output to each input.

This the capability of representing a changing input and using a single operator to calculate how the output evolves as the input evolves.

In our demo, we have the following explanation for the output. Below you will see what the difference traces are under Differences and the corresponding Versons are below.

Differences:
INPUT:
[]
[("hello", 2)]
[("hello", 2), ("world",-1)]
[("hello", 2), ("world",100)]
OUTPUT:
[]
[("hello", 1)]
[("hello", 0), ("world",0)]
[("hello", 0), ("world",1)]

Versions:
"INPUT VERSIONS"
[]
[("hello", 2)]
[("hello", 4),("world", -1)]

```
[("world", 99),("hello", 6)]
[("hello", 6),("world", 99)]
"OUTPUT VERSIONS"
[]
[("hello", 1)]
[("hello", 1),("world", 0)]
[("world", 1),("hello", 1)]
[("hello", 1),("world", 1)]
```

2)
The second demo shows the inner workings of the interpreter. Each Trace data type and Collection data type are represented using the modules Trace and Collection respectively. Go to the src file using `cd src`. Once you get there run `make build-test`. Next run `make unit-test`. You will see that all the unit tests pass.

Now if you go into the unit tests you will see some of the methods of Trace and Collections. You can search for `Demo1`, `Demo2`, `Demo3`, to get to the demo test cases. Comments should explain the exact details.

The purpose of these tests is to show the functionality of multiple dimensional traces that have not yet been applied to the language. (They were supposed to be used for the **loops** which would have added an extra dimension to an already preexisting trace).

Demo1:
This one shows what our Trace stores. It stores differences that we can get through the method get_diff_version.

Demo2:
This shows how to get a version. Our definition of less than is (t1, t2, t3…) <= (t1', t2', t3', …) iff (t1 <= t1'), (t2 <= t2'), ... In order to get the version we add the differences at all these indices. We use the method get_version.

Demo3:
This shows how to apply an operator. Distinct takes a input trace and outputs a trace. By applying the output trace to get_version (2,0), we get the same answer as applying distinct normally to version at (2,0) of the input trace.
For this one, we also encourage you to change [1;2] to [0;2]. The output becomes 1 because we no longer include the (This, -30).