Assignment 5

Exercise 1
Create the following simplified physical operators for your database system:

1. **Print**: Prints out all input tuples in a human-readable format.
2. **Table Scan**: Scans a relation and produces all tuples as output.
3. **Projection**: Projects to a subset of the input schema.
4. **Selection**: Implements predicates of the form $\bigwedge_i a_i = c_i$ where $a_i$ are attributes and $c_i$ are constants.
5. **Hash Join**: Compute inner join by storing left input in main memory, then find matches for each tuple from the right side. The predicate is of the form $left.a = right.b$.

In general, all operators should offer a superset of the following interface:

```cpp
void open() : Open the operator
bool next() : Produce the next tuple
vector<Register*> getOutput() : Get all produced values
void close() : Close the operator
```

Begin by creating a `Register` class that can be used to store and retrieve values of any type\(^1\) through methods like `int getInteger()` or `void setString(const string& s)`. It also needs to be able to compare `Register` objects (`operator<` and `operator==`) and compute a hash value (e.g. for `Hash Join` operators).

The `Table Scan` operator is initialized (in its constructor) with a relation. Its `next` method reads the next tuple (if any) and its `getOutput` method returns the values of the current tuple. The `Print` operator is initialized with an input operator and an output stream to which its `next` method writes the next tuple (if any) in a human-readable format. The `Projection` operator is initialized with an input operator and a list of register IDs (indexes into the register vector) it should project to. The `Selection` operator is initialized with an input operator, a register ID and a constant. The `Hash Join` operator is initialized with two input operators, and two register IDs. One ID is from the left side and one is from the right side.

---

\(^1\)In your implementation, you may restrict the database types to integer and a fixed-size character type.