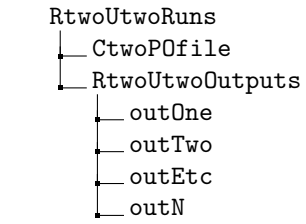
Assuming that you have a directory tree of this format:




(Sorry about the weird naming. The latex package I used doesn't allow numbers and dots in the names for some reason...)
This directory is created from running R2U2 on the MontyCarlo simulation of a timeline. The CtwoPOfile (contracts.c2po) has all of the contract names for the timeline runs. The RtwoUtwoOutputs has all of the outputs of each run of the timeline through R2U2.

**Algorithm 1** Creating Table Numbers from R2U2 Runs

**Input:** .c2po, folder containing outputs from r2u2 as .txt
**Output:** array containing aggregate data for multiple runs of r2u2

```
agg_data = empty_array;                                          ▷ 1
while contracts still in .c2po do
    agg_data[n][0] = contract_n                                  ▷ 2
end while

while file still in output folder do
    for line in .csv file do
        if agg_data[n][0] == contract_name then
            agg_data[n][1] += 1                                  ▷ 3
        end if
        if agg_data[n][0] == contract_name && failure then
            agg_data[n][2] += 1                                  ▷ 4
        end if
    end for
end while

for row in agg_data do
    agg_data[n][3] = (agg_data[2] ÷ agg_data[1]) × 100           ▷ 5
    agg_data[n][4] = (agg_data[2] ÷ total_runs) × 100            ▷ 6
end for

return return agg_data
```

Comments:

1. Initalize an empty array for storing data
2. Add contracts to first column of each row
3. Add the number of times a contract appears to the second column of a contract
4. If the line contains a failure for the contract name, add to third column
5. Add the failure vs instance for a contract
6. Add the failure vs total runs for a contract

How it Works:
The total number of runs is pulled from the total number of .csv's in the run folder.
The aggregate data .csv that the gui table pulls from will initally be created by the .c2po that has all of the contract names and requirements
If the .c2po file looks like this

```
INPUT
    var_1: type;

FTSPEC
    contract1_name: contract_1;
    contract2_name: contract_2;
    ...
    contractn_name: contract_n;
```

Then the aggregate data .csv file will look like this

| | | | | |
|---|---|---|---|---|
| contract_1 | 0 | 0 | 0 | 0 |
| contract_2 | 0 | 0 | 0 | 0 |
| ... | 0 | 0 | 0 | 0 |
| contract_n | 0 | 0 | 0 | 0 |

Table 1: The contract names will create the .csv with the following four values set initally to 0

When a contract appears in an individual run of R2U2, the "occurances" number will update. When a contract fails in an individual run of R2U2, the "failures" number will update.

Once all of the system runs have been processed by R2U2, the aggregate data .csv file will look like this

| | | | | |
|---|---|---|---|---|
| contract_1 | a | b | 0 | 0 |
| contract_2 | c | d | 0 | 0 |
| ... | ... | ... | 0 | 0 |
| contract_n | e | f | 0 | 0 |

Table 2: a,b,c,d,e,f are all positive integers

The last two rows are filled in by iterating down each line in the aggregate data .csv

| | | | | |
|---|---|---|---|---|
| contract_1 | a | b | (b ÷ a) × 100 | (b ÷ total_runs) × 100 |
| contract_2 | c | d | (d ÷ c) × 100 | (d ÷ total_runs) × 100 |
| ... | ... | ... | ... | ... |
| contract_n | e | f | (f ÷ e) × 100 | (f ÷ total_runs) × 100 |

Table 3: a,b,c,d,e,f are all positive integers, total_runs is the total number of runs for the system