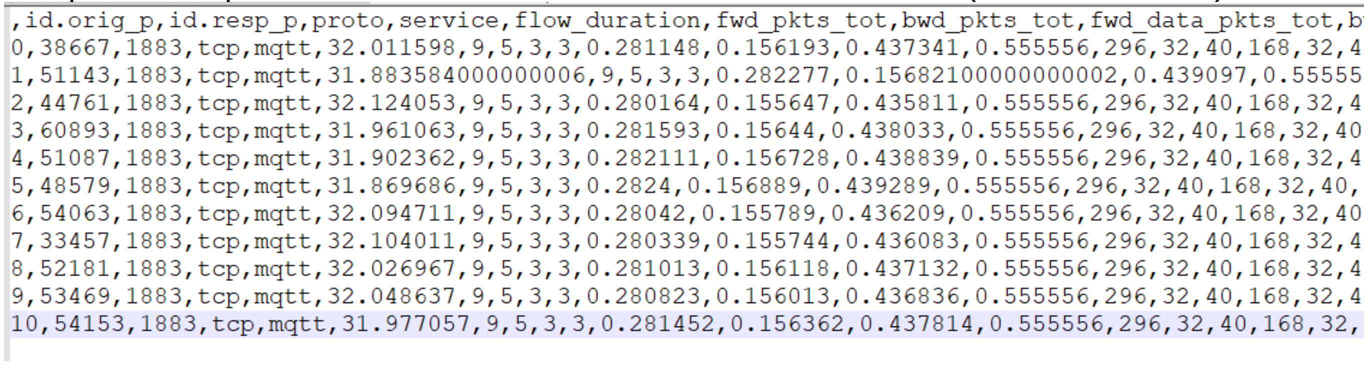
COEN 244 (Fall 2024) - Assignment 4 Description

Problem Statement One – File IO

In this assignment, we will process IoT data in a file. Please refer the dataset accessible from RT-IoT2022 - UCI Machine Learning Repository. In the report, please refer and cite the dataset as below.

B. S. and R. Nagapadma. "RT-IoT2022 ," UCI Machine Learning Repository, 2023. [Online]. Available: https://doi.org/10.24432/C5P338.

The partial samples are show blow. The first line contains attributes (aka column titles)



```
,id.orig_p,id.resp_p,proto,service,flow_duration,fwd_pkts_tot,bwd_pkts_tot,fwd_data_pkts_tot,b
0,38667,1883,tcp,mqtt,32.011598,9,5,3,3,0.281148,0.156193,0.437341,0.555556,296,32,40,168,32,4
1,51143,1883,tcp,mqtt,31.883584000000006,9,5,3,3,0.282277,0.15682100000000002,0.439097,0.55555
2,44761,1883,tcp,mqtt,32.124053,9,5,3,3,0.280164,0.155647,0.435811,0.555556,296,32,40,168,32,4
3,60893,1883,tcp,mqtt,31.961063,9,5,3,3,0.281593,0.15644,0.438033,0.555556,296,32,40,168,32,40
4,51087,1883,tcp,mqtt,31.902362,9,5,3,3,0.282111,0.156728,0.438839,0.555556,296,32,40,168,32,4
5,48579,1883,tcp,mqtt,31.869686,9,5,3,3,0.2824,0.156889,0.439289,0.555556,296,32,40,168,32,40,
6,54063,1883,tcp,mqtt,32.094711,9,5,3,3,0.28042,0.155789,0.436209,0.555556,296,32,40,168,32,40
7,33457,1883,tcp,mqtt,32.104011,9,5,3,3,0.280339,0.155744,0.436083,0.555556,296,32,40,168,32,4
8,52181,1883,tcp,mqtt,32.026967,9,5,3,3,0.281013,0.156118,0.437132,0.555556,296,32,40,168,32,4
9,53469,1883,tcp,mqtt,32.048637,9,5,3,3,0.280823,0.156013,0.436836,0.555556,296,32,40,168,32,4
10,54153,1883,tcp,mqtt,31.977057,9,5,3,3,0.281452,0.156362,0.437814,0.555556,296,32,40,168,32,
```

Applying a file stream, we can extract the data values line by line. For each line, we further apply string stream, we can extract token by token. We can then get a summary of how many attributes, and how many samples (one sample per line) the dataset.

The sample code below helps to print out the attribute names, # of attributes and each sample per line. The RT_IOT2022 contains the full dataset. To facilitate your testing and programming the code, a segment of the data is put in `iot-shard.txt` that contains less than 50 samples. It can be downloaded from the assignment specification files.

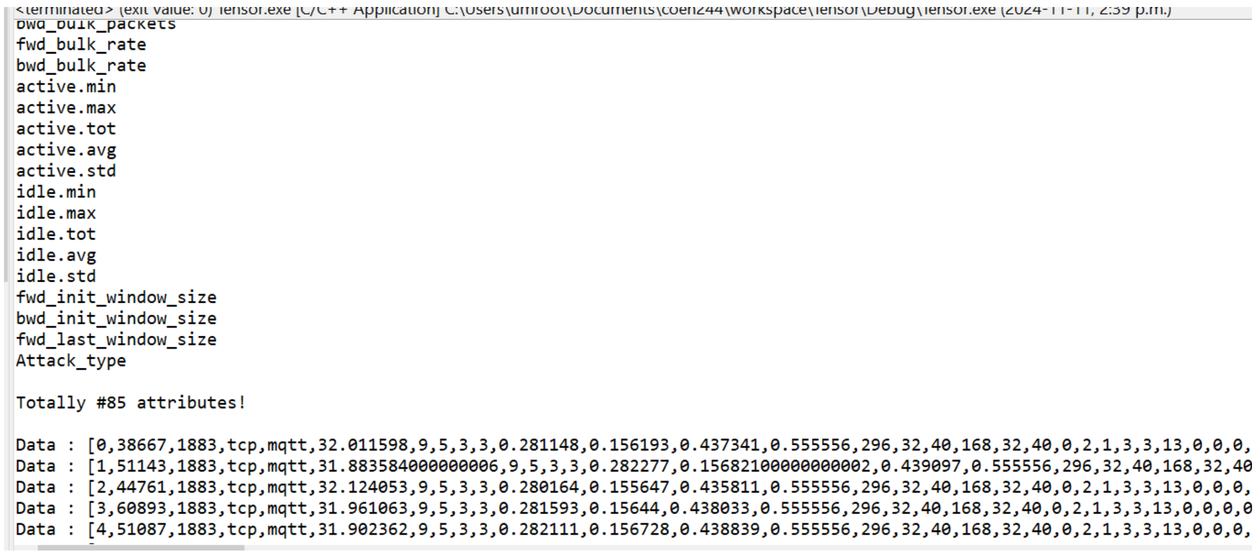
```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>


void testIoTDataSample(){
      string fileLocation = "RT_IOT2022";
```

```
        ifstream file(fileLocation);

        if(!file){
                string error_msg = fileLocation + " was not open properly!";
                throw std::runtime_error(error_msg);
        }


        string line;
        //get the first line that contains all the attribute' titles
        getline(file, line);

        //make a stringstream of the line to further tokenize each attribute
        stringstream lineString(line);

        int size(0);
        string token;
        //extract each token seperated by ','
        while (getline(lineString, token, ',')) {
                cout<<token<<endl;
                // count the number of attributes
                ++size;
        }
        cout << "\nTotally #" << size << " attributes!\n" <<endl;

//processing each line and store each line's value into one RankOneTensor object.
// when end_of_file is encountered, the while condition is false;
        while(getline(file, line)){
                lineString.clear();
                lineString.str("");
                lineString.str(line);
                string token;

                //Print out the data line by line using stringstream.
                cout << "Data : [";
        // stopping condition is that stringstream reaches to the end of the line;
                //eofbit is set for ss;
                while (lineString >> token) {
                        cout << token << "," ;
                }
                cout << " ]" << endl;
        }

        file.close();
}
```

The line below can be revised to the file location on your computer.

```
string fileLocation = "RT_IOT2022";
```

The expected output is given below.



```
<terminated> (exit value: 0) Tensor.exe [C/C++ Application] C:\Users\umroot\Documents\coen244\workspace\Tensor\Debug\Tensor.exe (2024-11-11, 2:39 p.m.)
bwd_bulk_packets
fwd_bulk_rate
bwd_bulk_rate
active.min
active.max
active.tot
active.avg
active.std
idle.min
idle.max
idle.tot
idle.avg
idle.std
fwd_init_window_size
bwd_init_window_size
fwd_last_window_size
Attack_type

Totally #85 attributes!

Data : [0,38667,1883,tcp,mqtt,32.011598,9,5,3,3,0.281148,0.156193,0.437341,0.555556,296,32,40,168,32,40,0,2,1,3,3,13,0,0,0,
Data : [1,51143,1883,tcp,mqtt,31.883584000000006,9,5,3,3,0.282277,0.15682100000000002,0.439097,0.555556,296,32,40,168,32,40
Data : [2,44761,1883,tcp,mqtt,32.124053,9,5,3,3,0.280164,0.155647,0.435811,0.555556,296,32,40,168,32,40,0,2,1,3,3,13,0,0,0,
Data : [3,60893,1883,tcp,mqtt,31.961063,9,5,3,3,0.281593,0.15644,0.438033,0.555556,296,32,40,168,32,40,0,2,1,3,3,13,0,0,0,0
Data : [4,51087,1883,tcp,mqtt,31.902362,9,5,3,3,0.282111,0.156728,0.438839,0.555556,296,32,40,168,32,40,0,2,1,3,3,13,0,0,0,
```

Task 1 : Program an application with the name as **DataSharding.cpp** with the main() function that partitions the original `RT_IOT2022` file and saves into a number of segments called data `shard.` A data shard does not have any overlapping with each other. The application can have a console prompt to ask for the size of a shard that is the number of samples (lines or rows) in each shard. Given the total number of samples, the number of shards should be upper bound of the total number of samples dividing the size. Each of the shard file should have the first row as the attributes or the column titles and be saved to a file name as `shard_<shard number>.txt`. Please replace <shard_number> as the actual index of your data segments. The directory can be given as another prompt input from the console.

The above code is provided as a sample to processing the data. It is not the direct solution to Task 1. You can leverage the whole sample code or reuse partial sample code or completely rewrite the program code to solve Task 1.

In addition to runnable application program code, please produce the screenshots of inputs to the application and the target directory with generated files of shards.

Submission artifacts:
Full program source code and the screenshots of inputs to the application and the target directory with generated files of shards.

Problem Statement Two – Template, Operator Overloading and Inheritance
In this problem, we will program the tensor template so that the element of tensor can be of generic type T instead of double values only. We can create a tensor for string data, or a tensor for double data or a tensor for class objects. The same concept of Tensor from Assignment 3 applies in this problem context.

The sample declaration is shown below:
```
template <typename T>
class RankOneTensorType : public BaseTensor{
private:
    std::vector<T> data;
public:
    RankOneTensorType() : data(0) {}
    RankOneTensorType(int size) : data(size) {}
    ~RankOneTensorType()=default;
    //other functions
}
```

The function `void loadData() override` is overridden due to the inheritance from the BaseTensor. Since initializing the tensor needs specific type, therefore the overriding function should be defined as function template. The program below shows the function template solutions for the specific type double and string.

```
template <>
void RankOneTensorType<double>::loadData() {
       std::fill(data.begin(), data.end(), valueGen());
}

template <>
void RankOneTensorType<string>::loadData() {
       std::fill(data.begin(), data.end(), std::to_string(valueGen()));
}
```

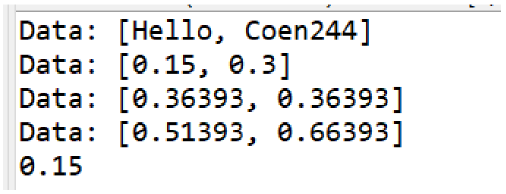
Task 2.1: Program the function templates for those operator overloading functions of `template <typename T>`
`class RankOneTensorType`. The same semantic meaning is defined in Assignment 3.

Operator	Semantic Meaning
++	Increment each element's value by 1
+	Add two tensors of the same dimension element wise; *invalid_argument should be thrown if two tensor objects' dimensions does not match*
+	Add one tensor with a scalar value element wise
=	Assign one tensor to another of the same dimension; *invalid_argument should be thrown if two tensor objects' dimensions does not match*
>>	Output the tensor's element values
<<	Input the tensor's element values by console input
[ ]	Retrieve value by index; *invalid_argument should be thrown if the*

	index is out of the range

Complete the full program code for `template <typename T> class RankOneTensorType.` In a separate `testDriver.cpp`, please add in the following unit test function. This unit test function can be refactored with other values. The sample output is given to illustrate the expectation of the result.

```
void testTensorType(){
	RankOneTensorType<string> tensor(2);
	tensor[0] = "Hello";
	tensor[1] = "Coen244";
	cout<< tensor;

	RankOneTensorType<double> dtensor(2);
	dtensor[0] = 0.15;
	dtensor[1] = 0.30;
	cout << dtensor;

	RankOneTensorType<double> dtensor2(2);
	dtensor2.loadData();

	cout << dtensor2;
	cout << dtensor+dtensor2;
	cout << dtensor[0];
}
```



```
Data: [Hello, Coen244]
Data: [0.15, 0.3]
Data: [0.36393, 0.36393]
Data: [0.51393, 0.66393]
0.15
```

Submission artifacts:
Full program source code and the screenshots of the output of running the test driver.

Task 2.2: A class IoTDataTensor further inherits from class `RankOneTensorType<string>.` loadData() overrides the function that reads data from the shard files produced in Task 1. loadData() function read the first line, extract tokes (i.e. column titles) and save the column titles in the private data member `std::vector<string> iot_category`. For each line of data samples, extract tokens and create one `RankOneTensorType<std::string> to store tokes.` Hence data samples in a shard file is stored in `std::vector<RankOneTensorType<std::string>> iot_data.`

```
class IoTDataTensor : public RankOneTensorType<string>{
public:

	IoTDataTensor (){
		iot_category.resize(0);
		iot_data.resize(0);
	}
```

```
        //get the value for at the entry of row and col
        double getValue(int row, int col);
        //get the attribute or column title at the index col
        string getCategory(int col);
        void loadData() override;
private:
        std::vector<string> iot_category;
        std::vector<RankOneTensorType<std::string>> iot_data;

};
```

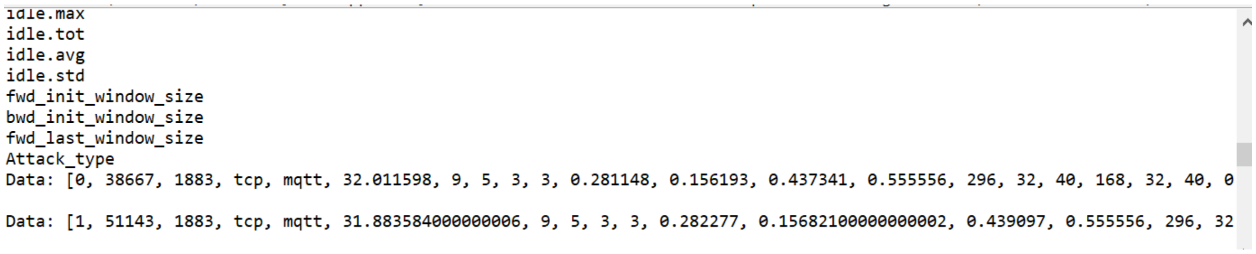
Complete the program solution of `class IoTDataTensor` for the three public member functions.

```
//get the value for at the entry of row and col
double getValue(int row, int col);
//get the attribute or column title at the index col
string getCategory(int col);
void loadData() override;
```

In a separate `testDriver.cpp`, please add in the following unit test function.

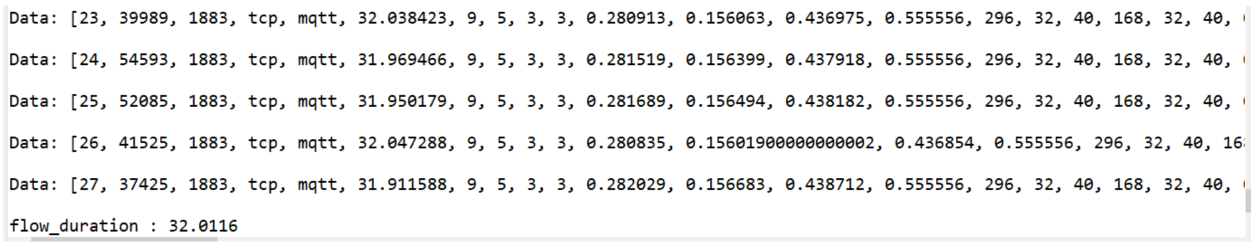
```
void testIoTDataTensor(){
        IoTDataTensor iot;
        try{
                iot.loadData();
                int row = 0;
                int col = 5;
                cout << iot.getCategory(col) << " : " << iot.getValue(row,col);

        }catch(std::runtime_error& e){
                std::cerr << e.what() << std::endl;
        }
}
```

This unit test function can be changed with other values. The sample output is given to illustrate the expectation of the result.



```
idle.max
idle.tot
idle.avg
idle.std
fwd_init_window_size
bwd_init_window_size
fwd_last_window_size
Attack_type
Data: [0, 38667, 1883, tcp, mqtt, 32.011598, 9, 5, 3, 3, 0.281148, 0.156193, 0.437341, 0.555556, 296, 32, 40, 168, 32, 40, 0

Data: [1, 51143, 1883, tcp, mqtt, 31.883584000000006, 9, 5, 3, 3, 0.282277, 0.15682100000000002, 0.439097, 0.555556, 296, 32
```

…



```
Data: [23, 39989, 1883, tcp, mqtt, 32.038423, 9, 5, 3, 3, 0.280913, 0.156063, 0.436975, 0.555556, 296, 32, 40, 168, 32, 40,

Data: [24, 54593, 1883, tcp, mqtt, 31.969466, 9, 5, 3, 3, 0.281519, 0.156399, 0.437918, 0.555556, 296, 32, 40, 168, 32, 40,

Data: [25, 52085, 1883, tcp, mqtt, 31.950179, 9, 5, 3, 3, 0.281689, 0.156494, 0.438182, 0.555556, 296, 32, 40, 168, 32, 40,

Data: [26, 41525, 1883, tcp, mqtt, 32.047288, 9, 5, 3, 3, 0.280835, 0.15601900000000002, 0.436854, 0.555556, 296, 32, 40, 16

Data: [27, 37425, 1883, tcp, mqtt, 31.911588, 9, 5, 3, 3, 0.282029, 0.156683, 0.438712, 0.555556, 296, 32, 40, 168, 32, 40,

flow_duration : 32.0116
```

Submission artifacts:
Full program source code and the screenshots of the output of running the test driver.

Submission Specification
In the report, please include the screenshots of Task 1 and Task 2.1 and Task 2.2 with clear descriptions each.
The submission should include all the source code and the report.pdf.

Create **[SID_1]_[SID_2]_A4.zip** (.gz, .tar, .zip are acceptable. .rar file is NOT acceptable) file contains a folder named with your student IDs.

The submission must be through Moodle on the submission link for Programming Assignment 4. Email submissions will NOT be accepted. All cases of plagiarism will be reported to authorities as per Concordia's plagiarism policy.

Task		Team members' development contribution
Task 1		
Task 2.1	Operator ++	
	Operator +	
	Operator +	
	Operator =	
	Operator <<	
	Operator >>	
	Operator [ ]	
	testDriver	
Task 2.2		

Assignment Marking Rubrics:

Level	Task 1 (20 marks); Task 2.1 operators and test driver (5 marks each); Task 2.2 (10 marks for each function).
5	All requirement addressed, programs run without errors in validation testing.
4	Missing 1 requirements, the rest of the function runs without errors.
3	Missing 2 requirements, the rest of the function runs without errors.
2	Missing more than 3 requirements, the program didn’t run properly.
1	Less than 2 requirements addressed, the program didn’t run properly.
0	The program is irrelevant to the solution