1. **How many of the returns were you able to process?**

214792 out of 475575 were not processed

2. **Show and interpret one explicit example of what you extracted from one tax return, including the text description before and after processing.**

Original:
THE ORGANIZATION SUPPORTS NORTH CAROLINA STATE UNIVERSITY BY OPERATING AN INVESTMENT FUND.

DocumentTermMatrix: ["an", "by", "carolina", "fund", "invest", "north", "oper", "organ", "state", "support", "the", "univers"]

Length: 12

TotalWorkersCount = 0

This was able to find the mission description and process it, with the worker count we received 0 which means it was able to find the counter correctly, if it could not find a count it would return -1 and discard this file.

3. **What are the dimensions of your term document matrix?**

(260783, 79653)

4. **How long did your program take to run? (Less than 30 minutes is easily attainable, but no problem if it takes longer, either.)**

Mine took about 44 minutes

5. **Which parts of the program took the longest time to run?**

I split my program into three parts, one that finds the files to collect, one that gets the data from the XML. The first part was the quickest, as expected, it only makes an array based on a directory. Then the second one, which is getting the elements from the XML, then third which was taking the array of descriptions and turning it into a single DTM.

```
using EzXML
using TextAnalysis
using Dates
using Serialization

# Global variables
dataDir = "2019"


function getFiles(dataDir)
    readdir(dataDir, join=true)
end

function getData(files)
    descriptions = String[]
    filesProcessed = 0
    filesNotProcessed= 0
    for file in files
        filesProcessed += 1
        # description
        description = getDescription(file)
        totalWorkers = getTotalWorkers(file)
        # println(totalWorkers)
        if description == ""
            # println(file * " description not processed")
            filesNotProcessed+=1
        elseif totalWorkers < 0
            # println(file * " workers not processed not processed")
            filesNotProcessed+=1
        else
            push!(descriptions, description)
        end
    end
    #! Need to print int correctly
    println(string(filesNotProcessed) * " out of " *
string(filesProcessed) * " were not processed")
    open("myfile.txt", "a") do io
        write(io, string(filesNotProcessed) * " out of " *
string(filesProcessed) * " were not processed")
```

```
        write(io, "\n")
    end
    return descriptions
end

function getDescription(file)
    doc = readxml(file)
    rootElement = root(doc)
    # Get the description
    missionDesc = findfirst("//MissionDesc", rootElement)
    if isnothing(missionDesc)
        missionDesc = findfirst("//PrimaryExemptPurposeTxt", rootElement)
    end

    if isnothing(missionDesc)
        return ""
    else
        return nodecontent(missionDesc)
    end
end

function getTotalWorkers(file)
    doc = readxml(file)
    rootElement = root(doc)
    # Get employees
    totalEmployees = findfirst("//EmployeeCnt", rootElement)
    # Get volunteers
    totalVolunteers = findfirst("//TotalVolunteersCnt", rootElement)

    if isnothing(totalEmployees) & isnothing(totalVolunteers)
        return -1
    elseif isnothing(totalVolunteers)
        totalEmployees = nodecontent(totalEmployees)
        return parse(Int64, totalEmployees)
    elseif isnothing(totalEmployees)
        totalVolunteers = nodecontent(totalVolunteers)
        return parse(Int64, totalVolunteers)
    else
        totalEmployees = nodecontent(totalEmployees)
        totalVolunteers = nodecontent(totalVolunteers)
```

```
        totalEmployees = parse(Int64, totalEmployees)
        totalVolunteers = parse(Int64, totalVolunteers)
        return totalEmployees + totalVolunteers
    end
end

function getDocumentTermMatrix(descriptions)
    sdList = StringDocument[]
    for description in descriptions
        sd = StringDocument(description)
        push!(sdList, sd)
    end
    c = Corpus(sdList)
    remove_case!(c)
    prepare!(c, strip_punctuation)
    stem!(c)
    update_lexicon!(c)

    d = DocumentTermMatrix(c)
    serialize("data", d)
end

open("myfile.txt", "a") do io
    write(io, Dates.format(now(), "HH:MM"))
    write(io, "\n")
end
@time files = getFiles(dataDir)
@time descriptions = getData(files)
@time getDocumentTermMatrix(descriptions)


open("myfile.txt", "a") do io
    write(io, Dates.format(now(), "HH:MM"))
    write(io, "\n")
end
```