A1:

To solve the above problem, I first went through the two resources 1 and 2 as mentioned on the assignment page as well as from the discussion in the class, I got to know that in order to access the twitter API and fetch URIs I need to get keys and tokens by creating a developer account on twitter.

Also, the program needed to install and import tweepy library which allows the developer to stream tweets based on key-words. The following dependencies were used:

* from tweepy.streaming import StreamListener
* from tweepy import OAuthHandler
* from tweepy import Stream
* import json
* import requests
* import urllib
* import tldextract
* from urllib.parse import urlparse, urljoin
* import sys

For this assignment I reused the code which was developed for assignment 2 which made implementation much easier and quicker. The program shown in Listing 1 was written in Python 3.5 for fetching 1000 unique links, omitting links from twitter domain. The following command was used to run and save the output in 1000ulinks.txt.

Program:

For my keywords I chose the ones which were currently trending on twitter. It first starts by listening to the stream of data from twitters streaming API, from which data is received in JSON format. The data was parsed using entities and urls object of a tweet and also filtered them for lengths and maximum characters. These links one by one were sent to getLinksFromTweet for further filtration.

In the next step, HTTP head and get requests[references] were made for each of the links fetched from the previous step and checked if expanded\_url entity had any redirects or were shortened for status 200 as well as status 301. Next I used the tdextract[reference] library for extracting the domain name as twitter since it had to be excluded. For checking the links to be unique I maintained two lists where in List 1 I stored all the URIs by formatting the query and parameters in links and List 2 where new links after comparing with List 1 , the links were again formed as original URIs. For this I used the urlparse and urljoin libraries[reference].

A2:

To solve the above problem, http://memgator.cs.odu.edu/timemap/json/

was used as mentioned in the question for getting the time maps for each of the 1000 URIs. I chose the JSON format to parse the data. The following dependencies were used:

* import requests
* import json
* import os
* import csv

First I checked the response code for first few URIs. From that I came to know about the links which had zero or no mementos, returned a response code of 404. So I made HTTP get and head requests to get the json data of timemaps and saved in .json file for URIs which returned 200 response using json.dump[reference] function and in .txt file as No-Mementos for those which returned 404 response.

Secondly to store the number of mementos and no of URIs in a csv file[reference] named timemaps.csv, I used a dictionary to arrange them with respect to Key as Memento\_count and URI\_Count as value.

The following program was written to implement the above problem in Python 3.5

Listing 2

There were few errors while plotting the histogram, for which I had to write a small piece of code shown below, for converting the timemaps.csv file into a single column file named counts.csv

Listing 3

I then created a simple histogram , using R[reference] shown shown in Listing 4, of URIs vs. Number of Mementos as shown in Figure 1. From the histogram observation we come to know that majority of URIs had zero or low counts of mementos.

A3

To solve the above problem, used the carbon dating tool[reference] provided in the question to retrieve the estimated creation dates and memgator[reference] to fetch the number of memtos for each of the URIs from the saved 1000ulinks.txt file. I first checked first few links and came to know that links with zero mementos either no creation date or had a creation date whereas URIs that had no date estimates didn't contain any mementos.

The following dependencies were used:

* import requests
* import json
* from datetime import datetime
* import csv

The program shown in Listing[] was written in Python 3.5 and the output was saved in dates.csv while with column names as Age and Memento\_Count.

Program listing

Also the program was founded the following values and saved in calc.csv file.

total URIs: 1000

\_ no mementos: 533

\_ no date estimate: 82

The following R[reference] program was used to create the scatterplot for Age in Days vs number of Mementos.