Stavros Stavroglou, ID: 201128565

1) **Middleware configuration**: I chose Storm standalone mode because I did not have access to a cluster in order to use a fully distributed mode for this assignment. The job ran on my laptop for 48 hours (RAM: 4GB , CPU: i5 Quadcore 2.40 GHz)

2) **Data Analytic Design**:

In my project I employed 1 spout and 3 bolts. In detail:

Spout: In the spout phase I created 2 successive tiers of tweet filtering. In the 1st tier, I filtered the incoming tweets using the "**Bounding Box**" [1] of the U.K. (see Appendix Fig. 1).

In the 2nd tier, I developed **3 ways** to identify the location of the tweet (unitary authority level, e.g. [2])

i. **Tweet Metadata (Geolocation)**: Extracted Latitude and Longitude of the tweet.

ii. **User Location**: In case no tweet metadata were embedded I extracted the user's profile location and checked if it matched with any of the U.K. cities stored in a list*.

iii. **Tweet Parsing**: In case neither tweet metadata nor user location were embedded, I parsed the tweet's text and checked if any token matched with any of the U.K. cities stored in a list*.

Cities' list*: see Appendix code 1.6.3

1st Bolt: ParseTweetBolt

Firstly, by using library "**OfflineReverseGeocode**" [3], after extracting the coordinates (by either of the 3 aforementioned ways) this bolt assigned every tweet in its corresponding unitary authority.
Secondly, by using a Java implementation [4] of the "**Aho-Corasick**" algorithm, this bolt searched for any of the **flu-words** (see code 1.6.2 and ref [5] for the list of flu-words) included in the tweet's text. If it identified one (flu-word) it assigned value 1 to the field "tweet-state", otherwise it assigned value 0. In either case it sends a tuple (including fields: "tweet-text","unit_authority_id", "tweet-state") to the 2nd Bolt.

2nd Bolt: TopUnitAuthoritiesBolt

Here, we employed 2 "**hashmap**" (key,value) structures.

i. **FluOccurences:** if tweet-state equals 1 it increased the value for the specfic unitary authority id (key).

ii. **TotalOccurences:** it increased the value for the specfic unitary authority id (key) regardless of the tweet-state.

Finally, it sent a tuple (including fields: "tweet-text", "unit_authority_id", "flu_occ", "total_occ") to the 3rd Bolt.

3rd Bolt: ReportBolt

This bolt set up the server that would handle receiving and sending messages. **Redis** [6] is an open source, BSD-licensed, key-value data store that also comes with a messaging system.

I used Redis to publish messages in this format: unit_authority_id DELIMITER flu_Occ DELIMITER total_Occ DELIMITER tweet-text.

Visualization

I used a Python script (see Appendix code 2.1) to establish a connection to the Redis server to receive messages from the 3rd Bolt. I used an open-source JavaScript library "**leaflet**" [7] for building interactive maps to visualise the ratio of flu related tweets in each unitary authority of the U.K. (see fig.2). I used the boundaries of U.K. counties from this reference: [8].

1) **Results**

I collected tweets from 380 Unitary Authorities (U.A.) for 48 hours, from 01:00 on the 16th of December until 01:00 on the 18th of December.

The results can best be summed in the following way:

- 78 out of 380 U.A. transmitted no tweets including any flu-words.
- 265 out of 380 U.A. transmitted flu-implying tweets at a percentage less than 5% of their total tweets.
- 20 out of 380 U.A. transmitted flu-implying tweets at a percentage between 5% and 20%
- 12 out of 380 U.A. transmitted flu-implying tweets at a percentage between 20% and 50%
- 5 out of 380 U.A. transmitted flu-implying tweets at a percentage between 50% and 91%

1) Discussion of Results

First of all as the results' summary and Fig.3 denote, only 37 out of 380 U.A. displayed a significant (>5%) percentage of flu occurrences. Notably the Top 5 U.A. suffering from flu are: Rhondda Cynon Taff (90.2%), Mid Sussex (81.4%), Rochford (72.1%), South Norfolk (57.9%) and Mid Suffolk (53.9%). The number of U.A.s with "serious" problem is small, nevertheless their extremely high percentage of flu occurrences could be alarming.

Secondly, if we take a closer look on the pure analogies (Fig.4) of the flu-related tweets to the total tweets per U.A. we observe that Rhondda Cynon Taff has by far the highest number both of total tweets and of flu-occurrences (total tweets = 1898, flu-occurrences = 1712). Probably in that U.A. there is a so-called "flu-season" or maybe even a flu outbreak. I would not dare to ring the same level of alarm for the rest of the U.A.s since their flu-occurrences are not that high in number see Fig.4.

Finally, I decided to use graph theory and construct the network of "which U.A. influenced which, in terms of flu contagion". In the network I created the nodes are the 380 U.A.s and the links among them are decided by a measure called Directed Cross Correlation coefficient [11]. Unlike simple Pearson Correlation Coefficient, the measure I used reveals the direction of contagion, so we can track from where did the flu spread to where. As we can see in Fig.5 there is a central cluster of U.A.s that are highly interrelated, while the majority lies in the periphery of the network and is disconnected. Judging by Fig.5 we can assess that some regions of the U.K. influence others and as a result there is a "contagion" effect taking place during the hours of my observation. (For a clearer picture of the network see image attached, because, due to the high quality of the output, the names of the U.A.s cannot be displayed properly.)

2) Conclusions and Recommendations

In my experiment I used Twitter Streaming API which accounts for just 2% [9] of the tweets being tweeted every moment. In order to undertake a similar task in a much larger scale I would have to gain access to Twitter Firehose API which provides 100% of the tweets being tweeted every moment, but it is available only after a mutual agreement with Twitter (and usually you have to pay a lot for such a privilege [8]). Probably I would also have to resort to Storm fully distributed mode (I would need a cluster to do so), in order to be capable of processing much larger amount of incoming data/second. Last but not least in order to have more useful conclusions I would run the project for far more than 48 hours, e.g. 1 year or even more.

REFERENCES

1. **https://www.udacity.com/course/real-time-analytics-with-apache-storm--ud381**
2. **http://boundingbox.klokantech.com/**
3. **http://mapit.mysociety.org/area/2564.html**
4. **https://github.com/AReallyGoodName/OfflineReverseGeocode**
5. **https://github.com/rripken/aho-corasick**
6. **http://www.lampos.net/sites/default/files/thesis/Lampos2012Thesis.pdf**
7. **https://en.wikipedia.org/wiki/Redis**
8. **http://leafletjs.com/**
9. **https://github.com/martinjc/UK-GeoJSON**
10. **https://www.echosec.net/twitter-api-vs-firehose/**
11. **Nie H. et al, 2015, Research of Chinese Stock Market' Complex Network Structure, International Journal of Economics and Finance; Vol. 7, No. 5**

APPENDIX

Fig.1 Bounding Box of the U.K. (below)

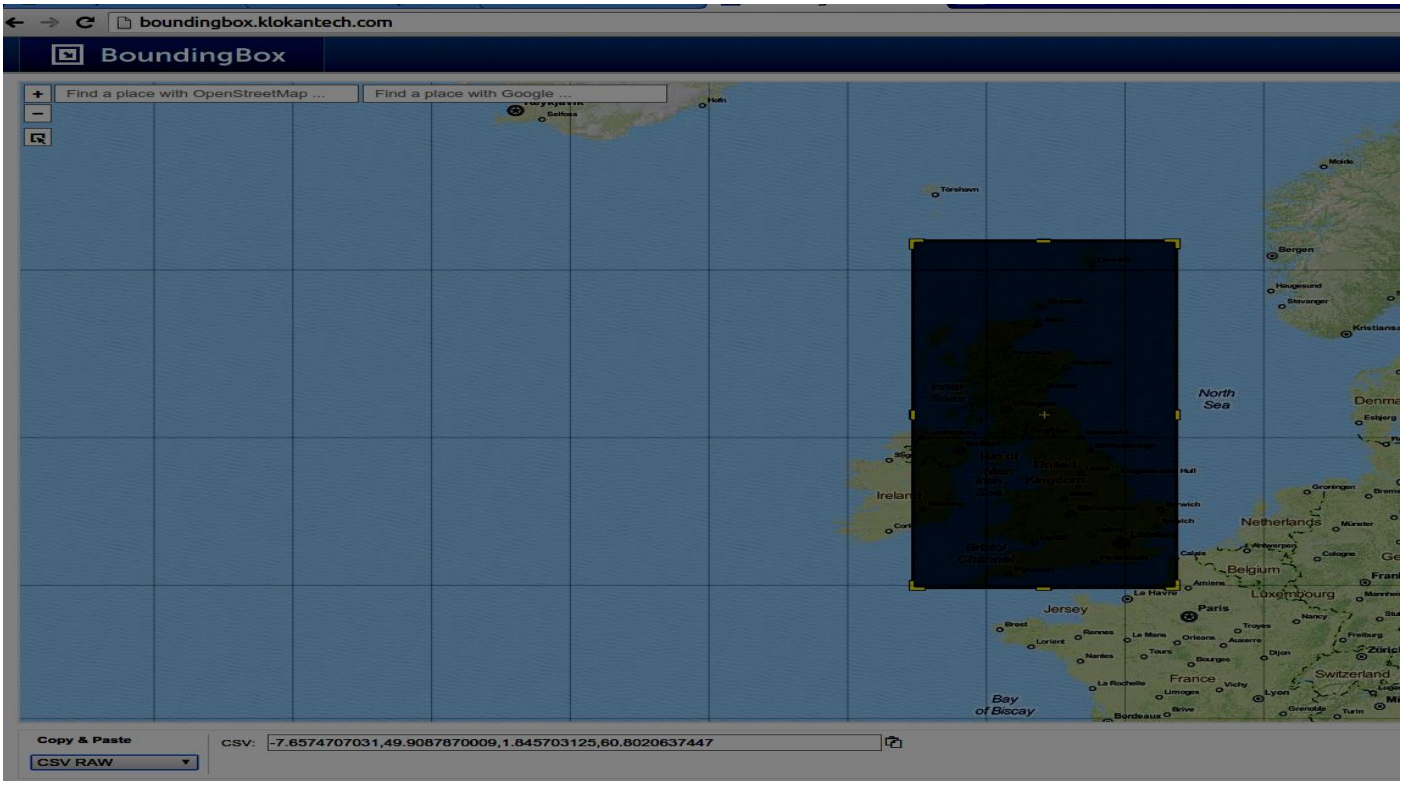


Fig.2 Sample screenshot taken at 01:22, on the 17th of December (below)

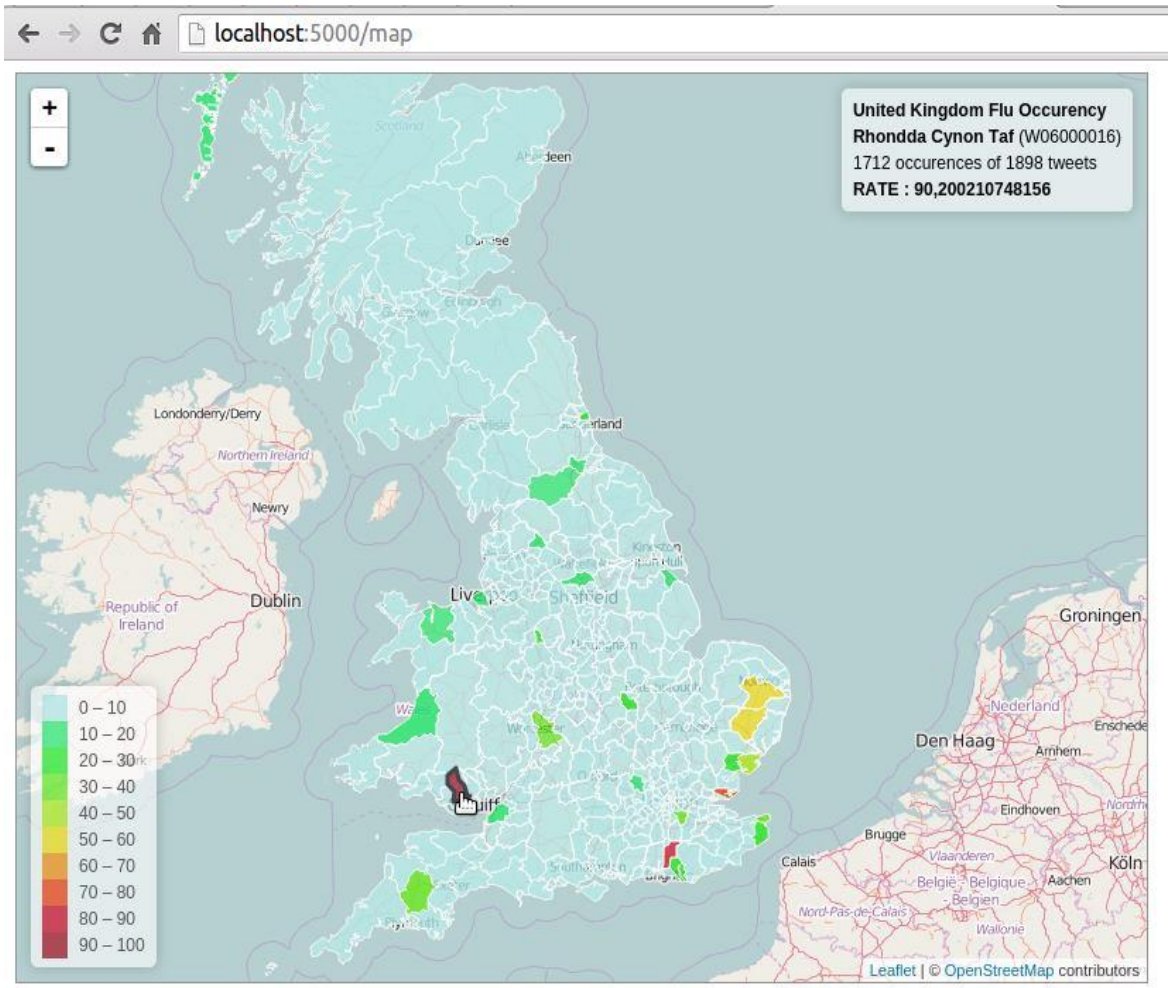


Fig.3 U.K. Unitary Authorities with more than 5% (of tweets) regarding flu occurrences

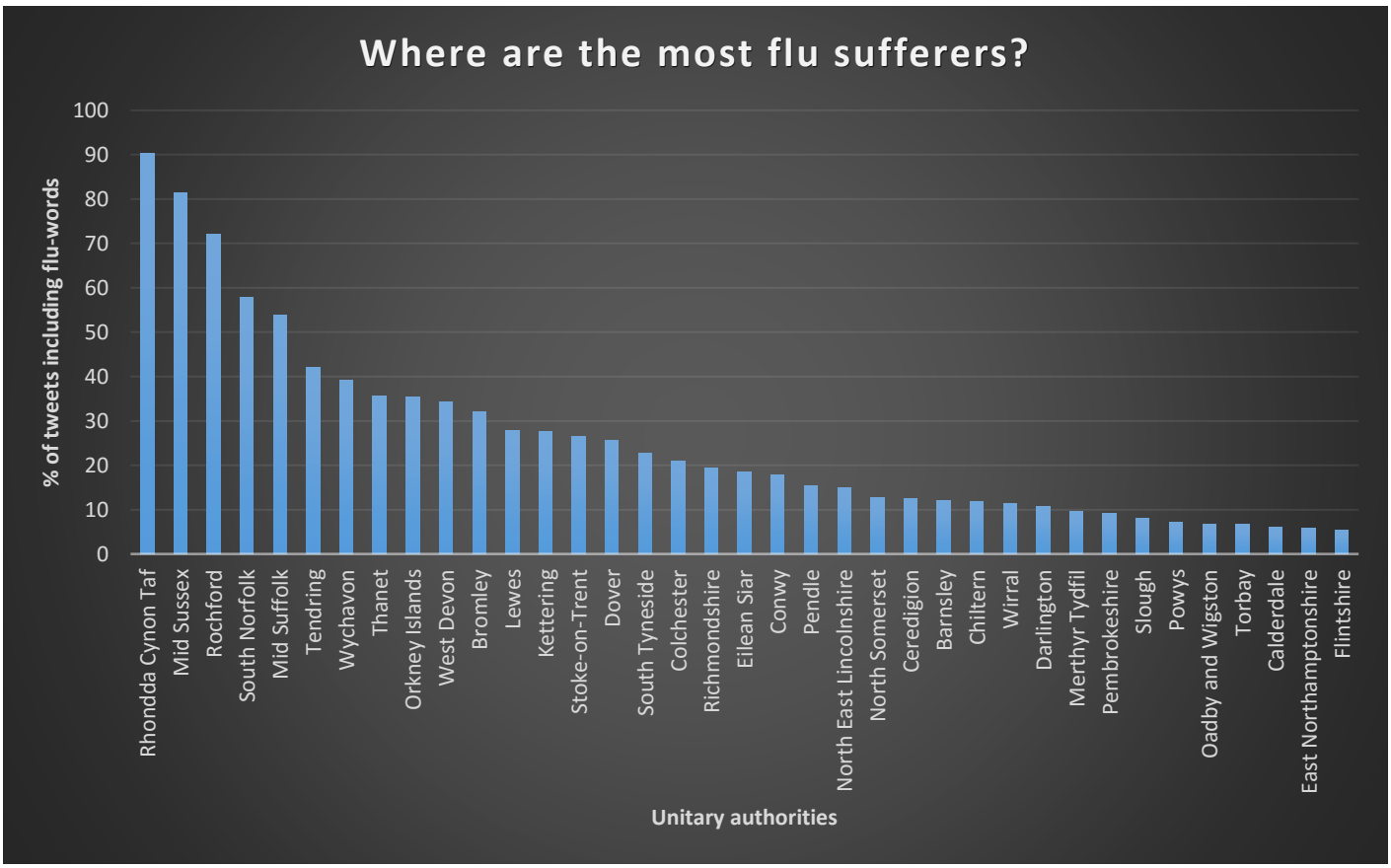


Fig.4 Clear analogies of the flu tweets/total tweets per Unitary Authority

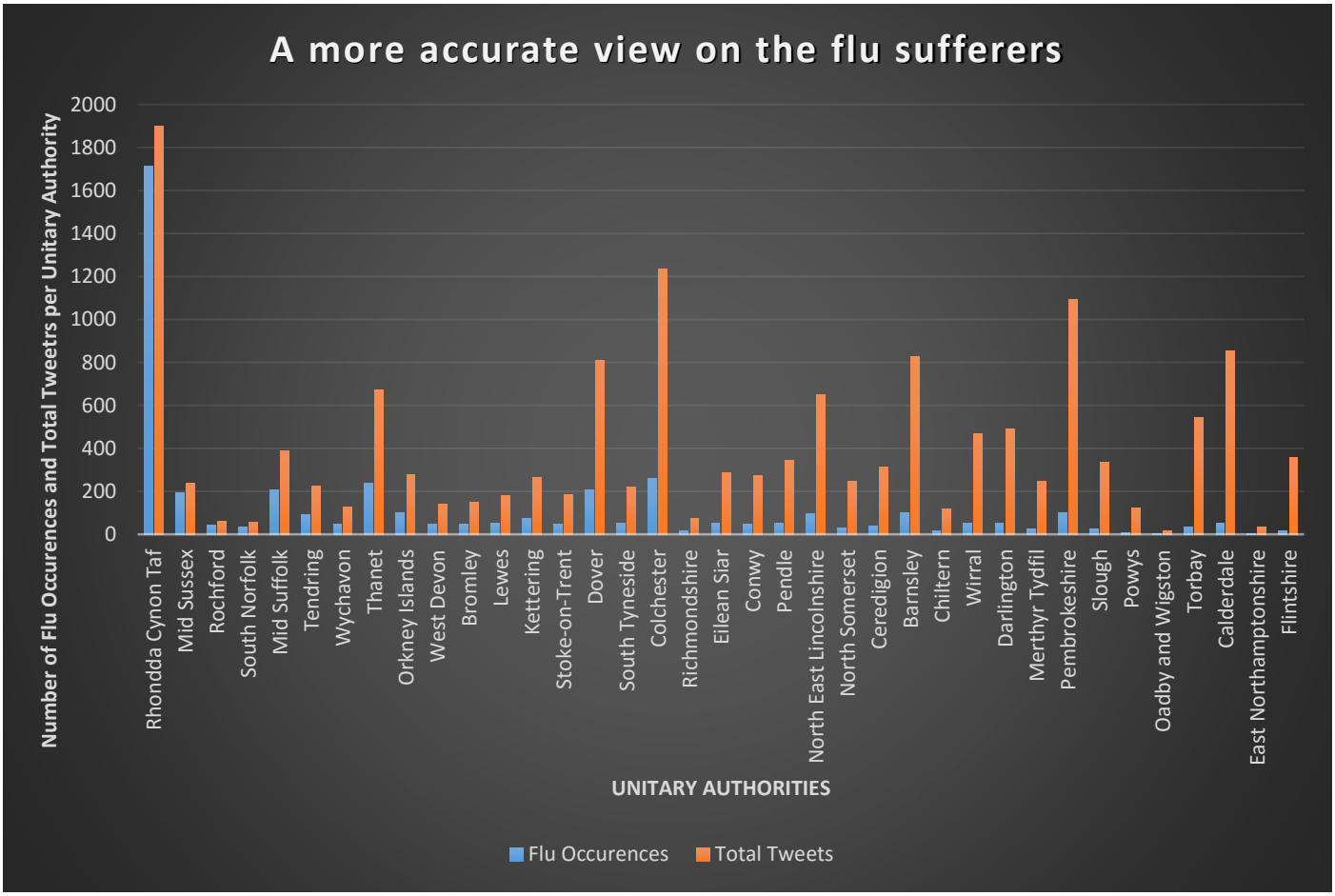
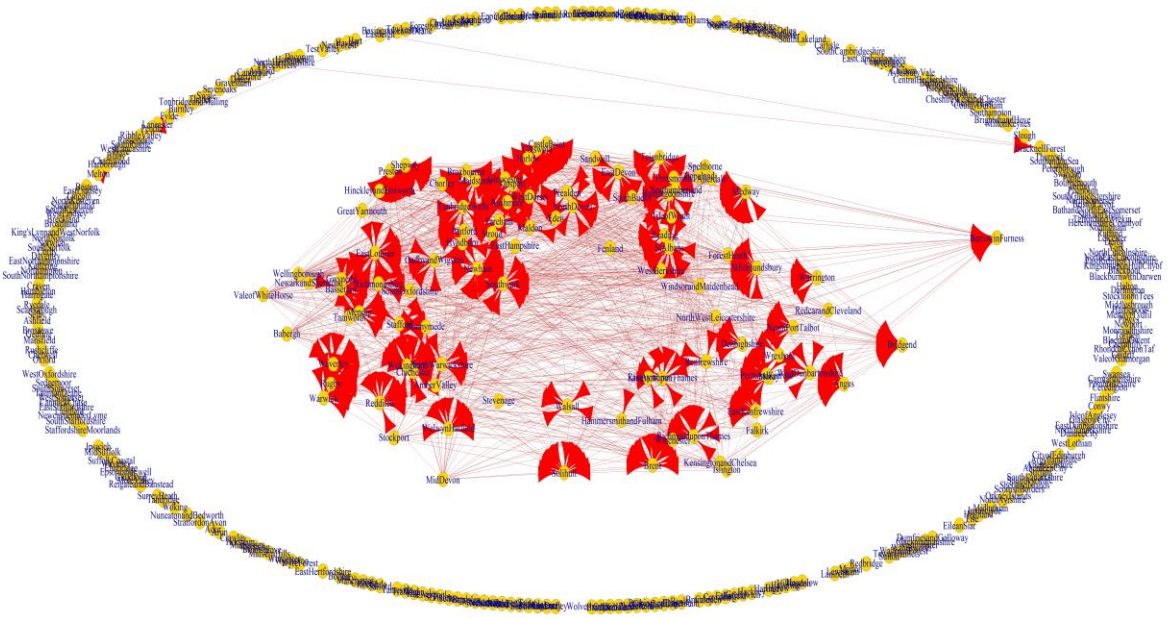


Fig.5 The Flu "contagion" network across the U.K. Unitary Authorities



CODES PROVIDED BELOW:

PART 1: STORM

Topology

1.1 Fludetector Topology (FluDetectorTopology.java)

```
package twitter.storm;

import backtype.storm.Config;
import backtype.storm.LocalCluster;
import backtype.storm.StormSubmitter;
import backtype.storm.topology.TopologyBuilder;
import backtype.storm.tuple.Fields;
import backtype.storm.utils.Utils;
import twitter.storm.bolt.ParseTweetBolt;
import twitter.storm.bolt.ReportBolt;
import twitter.storm.bolt.TopUnitAuthoritiesBolt;
import twitter.storm.spout.TweetSpout;

class FluDetectorTopology {

  public static void main(String[] args) throws Exception {

    // create the topology
    TopologyBuilder builder = new TopologyBuilder();

    // create the tweet spout with the credentials
    TweetSpout tweetSpout = new TweetSpout(
        "[Your customer key]",
        "[Your secret key]",
        "[Your access token]",
```

```
        "[Your access secret]"
    );

    // attach the tweet spout to the topology - parallelism of 1
    builder.setSpout("tweet-spout", tweetSpout, 1);

    // attach the parse tweet bolt using shuffle grouping
    builder.setBolt("parse-tweet-bolt", new ParseTweetBolt(), 10).shuffleGrouping("tweet-
spout");

    // attach the top unitary authorities bolt using fields grouping
    builder.setBolt("top-ua-bolt",      new TopUnitAuthoritiesBolt(),10).fieldsGrouping("parse-
tweet-bolt", new Fields("unit_authority_id"));

    // attach the report bolt using global grouping
    builder.setBolt("report-bolt",              new ReportBolt(),
1).globalGrouping("top-ua-bolt");

    // create the default configuration object
    Config conf = new Config();

    // set the configuration in debugging mode
    conf.setDebug(true);

    // run it in a live cluster
    if (args != null && args.length > 0) {

      // set the number of workers for running all spout and bolt tasks
      conf.setNumWorkers(3);

      // create the topology and submit with config
      StormSubmitter.submitTopology(args[0], conf, builder.createTopology());
    }
    // run it in a simulated local cluster
    else {

      // set the number of threads to run - similar to setting number of workers in live
cluster
      conf.setMaxTaskParallelism(4);

      // create the local cluster instance
      LocalCluster cluster = new LocalCluster();

      // submit the topology to the local cluster
      cluster.submitTopology("tweet-flu-count", conf, builder.createTopology());

      // let the topology run for 300 seconds. note topologies never terminate!
      Utils.sleep(300000000);

      // now kill the topology
      cluster.killTopology("tweet-flu-count");

      // we are done, shutdown the local cluster
      cluster.shutdown();
    }
  }
}
```

Spout

1.2 Tweet Spout (TweetSpout.java)

```
package twitter.storm.spout;

import backtype.storm.Config;
import backtype.storm.spout.SpoutOutputCollector;
import backtype.storm.task.TopologyContext;
import backtype.storm.topology.OutputFieldsDeclarer;
import backtype.storm.topology.base.BaseRichSpout;
import backtype.storm.tuple.Fields;
import backtype.storm.tuple.Values;
import backtype.storm.utils.Utils;
import twitter.storm.tools.CitiesLookup;
import twitter4j.conf.ConfigurationBuilder;
import twitter4j.FilterQuery;
import twitter4j.TwitterStream;
import twitter4j.TwitterStreamFactory;
import twitter4j.Status;
import twitter4j.StatusDeletionNotice;
import twitter4j.StatusListener;
import twitter4j.StallWarning;

import java.util.Map;
import java.util.concurrent.LinkedBlockingQueue;

/**
 * A spout that uses Twitter streaming API
 * for continuously getting tweets
 */
public class TweetSpout extends BaseRichSpout {

  private static final long serialVersionUID = -3480833811106377846L;

  // Twitter API authentication credentials
  String custkey, custsecret;
  String accesstoken, accesssecret;

  //To output tuples from spout to the next stage bolt
  SpoutOutputCollector collector;

  //Twitter4j - twitter stream to get tweets
  TwitterStream twitterStream;

  //Shared queue for getting buffering tweets received
  LinkedBlockingQueue<String> queue = null;

  // Holds a hash map of major uk cities
  CitiesLookup citieslookup;

  // Class for listening on the tweet stream - for twitter4j
  private class TweetListener implements StatusListener {

    // Implement the callback function when a tweet arrives
    @Override
    public void onStatus(Status status) {

        String geoInfo = "n/a";

        // Common delimiters
        String delims = "[\\s;.,:'!?()-]";

        // The location inside the user's profile
        String userLocation = status.getUser().getLocation();

        // if the tweet has geo metadata
        if(status.getGeoLocation() != null) {
```

```
                // get latitude and longitude
                geoInfo = String.valueOf(status.getGeoLocation().getLatitude()) +
                        "," + String.valueOf(status.getGeoLocation().getLongitude());

            // add the tweet to the queue
            queue.offer(status.getText() + "DELIMITER" + geoInfo);
        }
        // if the tweet does not have geo metadata
        // but the user filled the location field in his profile
        else if (userLocation != null) {

                // tokenize the location inside the user's profile
            String[] tokens = userLocation.split(delims);

            // for all tokens
            for (int i = 0; i < tokens.length; i++) {

                // check if this token it matches with the cities hashmap
                                if
(citieslookup.getCities().containsKey(tokens[i].trim().toUpperCase())) {

                                        // get the coordinates for this matched city
                                geoInfo =
citieslookup.getCities().get(tokens[i].trim().toUpperCase());

                                // add the tweet to the queue
                                queue.offer(status.getText() + "DELIMITER" + geoInfo);
                                break;
                                }
                        }
        }
        else {

                // tokenize the tweet text
            String[] tokens = status.getText().split(delims);

            // for all tokens
            for (int i = 0; i < tokens.length; i++) {

                // check if this token it matches with the cities hashmap
                                if (tokens[i].length() > 1 &&
citieslookup.getCities().containsKey(tokens[i].trim().toUpperCase())) {

                                        // get the coordinates for this matched city
                                geoInfo =
citieslookup.getCities().get(tokens[i].trim().toUpperCase());

                                // add the tweet to the queue
                                queue.offer(status.getText() + "DELIMITER" + geoInfo);
                                break;
                                }
                        }
        }
    }

    @Override
    public void onDeletionNotice(StatusDeletionNotice sdn) {

    }

    @Override
    public void onTrackLimitationNotice(int i) {

    }
```

```
    @Override
    public void onScrubGeo(long l, long l1) {

    }

    @Override
    public void onStallWarning(StallWarning warning) {

    }

    @Override
    public void onException(Exception e) {
      e.printStackTrace();
    }
  };

  /**
   * Constructor for tweet spout that accepts the twitter credentials
   */
  public TweetSpout(String key, String secret, String token, String tokensecret) {

          custkey = key;
          custsecret = secret;
          accesstoken = token;
          accesssecret = tokensecret;
  }

  @Override
  public void open(Map map, TopologyContext topologyContext, SpoutOutputCollector
spoutOutputCollector) {

        // create the buffer to block tweets
    queue = new LinkedBlockingQueue<String>(1000);

    // initialize the hashmap of major uk cities
    citieslookup = new CitiesLookup();

    // save the output collector for emitting tuples
    collector = spoutOutputCollector;

    // build the config with credentials for twitter 4j
    ConfigurationBuilder config = new ConfigurationBuilder()
                                                        .setOAuthConsumerKey(custkey)
                                                        .setOAuthConsumerSecret(custsecre
t)
                                                        .setOAuthAccessToken(accesstoken)
                                                        .setOAuthAccessTokenSecret(access
secret);

    // create the twitter stream factory with the configuration
    TwitterStreamFactory fact = new TwitterStreamFactory(config.build());

    // get an instance of twitter stream
    twitterStream = fact.getInstance();

    // get an instance of FilterQuery
    FilterQuery filterQuery = new FilterQuery();

    // UK Bounding Box filter
    filterQuery.locations(new double[][] {
                                            new double[]{-14.0155172348,49.6739997864},
                                            new double[]{2.0919117928,61.061000824}
                                        });

    // Language filter
```

```
  String[] lang = {"en"};
  filterQuery.language(lang);

  twitterStream.addListener(new TweetListener());
  twitterStream.firehose(0);

  // add the filter to the stream
  twitterStream.filter(filterQuery);

}

@Override
public void nextTuple()  {

  // try to pick a tweet from the buffer
  String tweet = queue.poll();

  String geoInfo;
  String tweetText;

  // if no tweet is available, wait for 50 ms and return
  if (tweet == null) {
    Utils.sleep(50);
    return;
  }
  else {
      // get the tweet text
      tweetText = tweet.split("DELIMITER")[0];
      // get the tweet geospatial info (lat/long)
      geoInfo = tweet.split("DELIMITER")[1];
  }

  if (geoInfo != null && !geoInfo.equals("n/a")) {

      // emit the tweet to the next bolt
      collector.emit(new Values(tweet));
  }

}

@Override
public void close() {

      // shutdown the stream - when we are going to exit
  twitterStream.shutdown();
}

/**
 * Component specific configuration
 */
@Override
public Map<String, Object> getComponentConfiguration() {

  Config conf = new Config();  // create the component config
  conf.setMaxTaskParallelism(1);  // set the parallelism for this spout to be 1
  return conf;
}

@Override
public void declareOutputFields(OutputFieldsDeclarer outputFieldsDeclarer) {

  // tell storm the schema of the output tuple for this spout
  // tuple consists of a single column called 'tweet'
  outputFieldsDeclarer.declare(new Fields("tweet"));
}
```

```
}
```

Bolts

1.3 Parse Tweet Bolt (ParseTweetBolt.java)

```
package twitter.storm.bolt;

import backtype.storm.task.OutputCollector;
import backtype.storm.task.TopologyContext;
import backtype.storm.topology.OutputFieldsDeclarer;
import backtype.storm.topology.base.BaseRichBolt;
import backtype.storm.tuple.Fields;
import backtype.storm.tuple.Tuple;
import backtype.storm.tuple.Values;
import twitter.storm.tools.TrackKeywords;
import twitter.storm.tools.UnitaryAuthoritiesLookup;

import java.util.Map;
import org.ahocorasick.trie.Token;
import java.util.Collection;

/**
 * A bolt that parses the incoming tweet
 */
public class ParseTweetBolt extends BaseRichBolt {

  private static final long serialVersionUID = -676893541771976399L;

  // To output tuples from this bolt to the TopUnitAuthorities bolt
  OutputCollector collector;

  // holds the hashmap of all uk unitary authorities
  UnitaryAuthoritiesLookup ualookup;

  // class to track words
  TrackKeywords trackWords;

  @Override
  public void prepare(Map map, TopologyContext topologyContext, OutputCollector
outputCollector) {

    // save the output collector for emitting tuples
    collector = outputCollector;

    // initialize the hashmap of all uk unitary authorities
    ualookup = new UnitaryAuthoritiesLookup();

    // create an instance to track words
    trackWords = new TrackKeywords();

  }

  // Given in iput a tweet check if it matches with any defined track words
  public boolean doesTweetMatch(String tweet) {
```

```
            // tokenize the tweet text
        Collection<Token> tokens = trackWords.getSearchTrie().tokenize(tweet);

        // for all tokens
        for (Token token : tokens) {
            // if the tweet matches return true
            if (token.isMatch()) {
                    return true;
            }
        }

        return false;
    }

    @Override
    public void execute(Tuple tuple) {

          // get fields from the incoming tuple
      String tweet = tuple.getStringByField("tweet").split("DELIMITER")[0];
      double latitude =
Double.parseDouble(tuple.getStringByField("tweet").split("DELIMITER")[1].split(",")[0]);
      double longitude =
Double.parseDouble(tuple.getStringByField("tweet").split("DELIMITER")[1].split(",")[1]);

      // assign a unitary authority id for these coordinates (latitude,longitude)
      String unit_authority_id = ualookup.getUACodeByGeo(latitude, longitude);

      // if the tweet contains at least one defined keyword emit it with the state 1
      if (doesTweetMatch(tweet)) {
          collector.emit(new Values(tweet,unit_authority_id,1));
      }
      // if the tweet does NOT contain any defined keyword emit the output tuple with the state 0
      else {
          collector.emit(new Values(tweet,unit_authority_id,0));
      }

    }

    @Override
    public void declareOutputFields(OutputFieldsDeclarer declarer) {

      // tell storm the schema of the output tuple for this bolt
      declarer.declare(new Fields("tweet-text","unit_authority_id","tweet-state"));

    }

}
```

1.4 Top Unit Authorities Bolt (TopUnitAuthoritiesBolt.java)

```
package twitter.storm.bolt;

import backtype.storm.topology.OutputFieldsDeclarer;

import backtype.storm.topology.base.BaseRichBolt;
import backtype.storm.tuple.Fields;
import backtype.storm.tuple.Tuple;
import backtype.storm.tuple.Values;
import backtype.storm.task.TopologyContext;
import backtype.storm.task.OutputCollector;

import java.util.HashMap;
```

```
import java.util.Map;

/**
 * A bolt that counts the flu and total occurences
 * for all the UK unitary authorities
 */
public class TopUnitAuthoritiesBolt extends BaseRichBolt {

        private static final long serialVersionUID = 5905712870882061066L;

        // To output tuples from this bolt to the Report bolt
        private OutputCollector collector;

        // HashMap that counts for every unitary authority the flu occurences
        private Map<String, Integer> fluOccurences;

        // HashMap that counts for every unitary authority the total occurences
        private Map<String, Integer> totalOccurences;

        @Override
        public void prepare(Map map, TopologyContext topologyContext, OutputCollector
outputCollector) {

                // save the output collector for emitting tuples
                collector = outputCollector;

                // initialize the hashmaps
                fluOccurences = new HashMap<String, Integer>();
                totalOccurences = new HashMap<String, Integer>();
        }

        public void execute(Tuple tuple) {

                // get fields from the incoming tuple
                String tweet = tuple.getStringByField("tweet-text");
                String unit_authority_id = tuple.getStringByField("unit_authority_id");
                int tweetState = tuple.getIntegerByField("tweet-state");

                // if the tweet state is 1 (the tweet contains flu words)
                if (tweetState == 1) {

                        // increment the fluOccurences value for this unitary authority
                        if (fluOccurences.get(unit_authority_id) == null)
{  fluOccurences.put(unit_authority_id,1); }
                        else { fluOccurences.put(unit_authority_id,
fluOccurences.get(unit_authority_id) + 1); }

                }
                // if the tweet state is 0 (the tweet does NOT contain flu words)
                else if (tweetState == 0) {

                        // initialize the fluOccurences value for this unitary authority
                        if (fluOccurences.get(unit_authority_id) == null)
{ fluOccurences.put(unit_authority_id,0); }

                }

                // increment the totalOccurences value for this unitary authority
                if (totalOccurences.get(unit_authority_id) == null)
{ totalOccurences.put(unit_authority_id,1); }
                else { totalOccurences.put(unit_authority_id,
totalOccurences.get(unit_authority_id) + 1); }

                // emit the output tuple for this bolt
                collector.emit(new
```

```
Values(tweet,unit_authority_id,fluOccurences.get(unit_authority_id),totalOccurences.get(unit_au
thority_id)));
        }

        @Override
        public void declareOutputFields(OutputFieldsDeclarer outputFieldsDeclarer) {

                // tell storm the schema of the output tuple for this bolt
                outputFieldsDeclarer.declare(new Fields("tweet-
text","unit_authority_id","flu_occ","all_occ"));
        }

}
```

1.5 Report Bolt (ReportBolt.java)

```
package twitter.storm.bolt;

import backtype.storm.task.OutputCollector;
import backtype.storm.task.TopologyContext;
import backtype.storm.topology.OutputFieldsDeclarer;
import backtype.storm.topology.base.BaseRichBolt;
import backtype.storm.tuple.Tuple;
import java.util.Map;

import com.lambdaworks.redis.RedisClient;
import com.lambdaworks.redis.RedisConnection;

/**
 * A bolt that prints messages to redis
 */
public class ReportBolt extends BaseRichBolt {

  private static final long serialVersionUID = 532484535287440101L;

  // place holder to keep the connection to redis
  transient RedisConnection<String,String> redis;

  @Override
  public void prepare(Map map, TopologyContext topologyContext, OutputCollector
outputCollector) {

        // instantiate a redis connection
    RedisClient client = new RedisClient("localhost", 6379);

    // initiate the actual connection
    redis = client.connect();
  }

  @Override
  public void execute(Tuple tuple) {

        // get fields from the incoming tuple
        String tweet = tuple.getStringByField("tweet-text");
    String unit_authority_id = tuple.getStringByField("unit_authority_id");
    int fluOccurences = tuple.getIntegerByField("flu_occ");
    int allOccurences = tuple.getIntegerByField("all_occ");

    // publish a message to redis using the channel FluDetectorTopology
    redis.publish("FluDetectorTopology", unit_authority_id + "DELIMITER" +
                                                                    fluOccurences +
```

```
"DELIMITER" +
                                                                    allOccurences +
"DELIMITER" +
                                                                    tweet
                        );
  }

  public void declareOutputFields(OutputFieldsDeclarer declarer)  {

    // nothing to add - since it is the final bolt

  }
}
```

1.6 Tools

1.6.1 Unitary Authorities Lookup (UnitaryAuthoritiesLookup.java)

```
package twitter.storm.tools;

import java.io.Serializable;
import java.util.HashMap;

import geocode.ReverseGeoCode;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

/**
 * Class that initializes a HashMap
 * of UK unitary authority names as keys and
 * their codes as values
 */
public class UnitaryAuthoritiesLookup implements Serializable {

        private static final long serialVersionUID = -9102878650001058090L;

        // UK Counties data
        private HashMap<String, String> data;

        // file to read the full data
        String csvFile = "geoinfo_uk.csv";

        FileInputStream fis;
        ReverseGeoCode rgc;

        // given in input a county name return the id code
        public int getGeoID(String county){

                if (data.containsKey(county.toUpperCase())) {
                        return Integer.parseInt(data.get(county.toUpperCase()));
                }

                return 0;
        }

        // given in input the coordinates of a point return the appropriate unitary authority
code
```

```
public String getUACodeByGeo(double latitude, double longitude){

        return rgc.nearestPlace(latitude, longitude).geoid;
}

public UnitaryAuthoritiesLookup() {

        try {

                fis = new FileInputStream(csvFile);

        } catch (FileNotFoundException e) {

                e.printStackTrace();
        }

        try {

                rgc = new ReverseGeoCode(fis, false);

        } catch (IOException e) {

                e.printStackTrace();
        }

        data = new HashMap<String, String>();

        // UNITED KINGDOM UNITARY AUTHORITIES

        data.put("Hartlepool".toUpperCase(),"cE06000001");
        data.put("Middlesbrough".toUpperCase(),"cE06000002");
        data.put("Redcar and Cleveland".toUpperCase(),"cE06000003");
        data.put("Stockton-on-Tees".toUpperCase(),"cE06000004");
        data.put("Darlington".toUpperCase(),"cE06000005");
        data.put("Halton".toUpperCase(),"cE06000006");
        data.put("Warrington".toUpperCase(),"cE06000007");
        data.put("Blackburn with Darwen".toUpperCase(),"cE06000008");
        data.put("Blackpool".toUpperCase(),"cE06000009");
        data.put("Kingston upon Hull, City of".toUpperCase(),"cE06000010");
        data.put("East Riding of Yorkshire".toUpperCase(),"cE06000011");
        data.put("North East Lincolnshire".toUpperCase(),"cE06000012");
        data.put("North Lincolnshire".toUpperCase(),"cE06000013");
        data.put("York".toUpperCase(),"cE06000014");
        data.put("Derby".toUpperCase(),"cE06000015");
        data.put("Leicester".toUpperCase(),"cE06000016");
        data.put("Rutland".toUpperCase(),"cE06000017");
        data.put("Nottingham".toUpperCase(),"cE06000018");
        data.put("Herefordshire, County of".toUpperCase(),"cE06000019");
        data.put("Telford and Wrekin".toUpperCase(),"cE06000020");
        data.put("Stoke-on-Trent".toUpperCase(),"cE06000021");
        data.put("Bath and North East Somerset".toUpperCase(),"cE06000022");
        data.put("Bristol, City of".toUpperCase(),"cE06000023");
        data.put("North Somerset".toUpperCase(),"cE06000024");
        data.put("South Gloucestershire".toUpperCase(),"cE06000025");
        data.put("Plymouth".toUpperCase(),"cE06000026");
        data.put("Torbay".toUpperCase(),"cE06000027");
        data.put("Bournemouth".toUpperCase(),"cE06000028");
        data.put("Poole".toUpperCase(),"cE06000029");
        data.put("Swindon".toUpperCase(),"cE06000030");
        data.put("Peterborough".toUpperCase(),"cE06000031");
        data.put("Luton".toUpperCase(),"cE06000032");
        data.put("Southend-on-Sea".toUpperCase(),"cE06000033");
        data.put("Thurrock".toUpperCase(),"cE06000034");
        data.put("Medway".toUpperCase(),"cE06000035");
        data.put("Bracknell Forest".toUpperCase(),"cE06000036");
```

```
data.put("West Berkshire".toUpperCase(),"cE06000037");
data.put("Reading".toUpperCase(),"cE06000038");
data.put("Slough".toUpperCase(),"cE06000039");
data.put("Windsor and Maidenhead".toUpperCase(),"cE06000040");
data.put("Wokingham".toUpperCase(),"cE06000041");
data.put("Milton Keynes".toUpperCase(),"cE06000042");
data.put("Brighton and Hove".toUpperCase(),"cE06000043");
data.put("Portsmouth".toUpperCase(),"cE06000044");
data.put("Southampton".toUpperCase(),"cE06000045");
data.put("Isle of Wight".toUpperCase(),"cE06000046");
data.put("County Durham".toUpperCase(),"cE06000047");
data.put("Cheshire East".toUpperCase(),"cE06000049");
data.put("Cheshire West and Chester".toUpperCase(),"cE06000050");
data.put("Shropshire".toUpperCase(),"cE06000051");
data.put("Cornwall".toUpperCase(),"cE06000052");
data.put("Isles of Scilly".toUpperCase(),"cE06000053");
data.put("Wiltshire".toUpperCase(),"cE06000054");
data.put("Bedford".toUpperCase(),"cE06000055");
data.put("Central Bedfordshire".toUpperCase(),"cE06000056");
data.put("Northumberland".toUpperCase(),"cE06000057");
data.put("Aylesbury Vale".toUpperCase(),"cE07000004");
data.put("Chiltern".toUpperCase(),"cE07000005");
data.put("South Bucks".toUpperCase(),"cE07000006");
data.put("Wycombe".toUpperCase(),"cE07000007");
data.put("Cambridge".toUpperCase(),"cE07000008");
data.put("East Cambridgeshire".toUpperCase(),"cE07000009");
data.put("Fenland".toUpperCase(),"cE07000010");
data.put("Huntingdonshire".toUpperCase(),"cE07000011");
data.put("South Cambridgeshire".toUpperCase(),"cE07000012");
data.put("Allerdale".toUpperCase(),"cE07000026");
data.put("Barrow-in-Furness".toUpperCase(),"cE07000027");
data.put("Carlisle".toUpperCase(),"cE07000028");
data.put("Copeland".toUpperCase(),"cE07000029");
data.put("Eden".toUpperCase(),"cE07000030");
data.put("South Lakeland".toUpperCase(),"cE07000031");
data.put("Amber Valley".toUpperCase(),"cE07000032");
data.put("Bolsover".toUpperCase(),"cE07000033");
data.put("Chesterfield".toUpperCase(),"cE07000034");
data.put("Derbyshire Dales".toUpperCase(),"cE07000035");
data.put("Erewash".toUpperCase(),"cE07000036");
data.put("High Peak".toUpperCase(),"cE07000037");
data.put("North East Derbyshire".toUpperCase(),"cE07000038");
data.put("South Derbyshire".toUpperCase(),"cE07000039");
data.put("East Devon".toUpperCase(),"cE07000040");
data.put("Exeter".toUpperCase(),"cE07000041");
data.put("Mid Devon".toUpperCase(),"cE07000042");
data.put("North Devon".toUpperCase(),"cE07000043");
data.put("South Hams".toUpperCase(),"cE07000044");
data.put("Teignbridge".toUpperCase(),"cE07000045");
data.put("Torridge".toUpperCase(),"cE07000046");
data.put("West Devon".toUpperCase(),"cE07000047");
data.put("Christchurch".toUpperCase(),"cE07000048");
data.put("East Dorset".toUpperCase(),"cE07000049");
data.put("North Dorset".toUpperCase(),"cE07000050");
data.put("Purbeck".toUpperCase(),"cE07000051");
data.put("West Dorset".toUpperCase(),"cE07000052");
data.put("Weymouth and Portland".toUpperCase(),"cE07000053");
data.put("Eastbourne".toUpperCase(),"cE07000061");
data.put("Hastings".toUpperCase(),"cE07000062");
data.put("Lewes".toUpperCase(),"cE07000063");
data.put("Rother".toUpperCase(),"cE07000064");
data.put("Wealden".toUpperCase(),"cE07000065");
data.put("Basildon".toUpperCase(),"cE07000066");
data.put("Braintree".toUpperCase(),"cE07000067");
data.put("Brentwood".toUpperCase(),"cE07000068");
```

```
data.put("Castle Point".toUpperCase(),"cE07000069");
data.put("Chelmsford".toUpperCase(),"cE07000070");
data.put("Colchester".toUpperCase(),"cE07000071");
data.put("Epping Forest".toUpperCase(),"cE07000072");
data.put("Harlow".toUpperCase(),"cE07000073");
data.put("Maldon".toUpperCase(),"cE07000074");
data.put("Rochford".toUpperCase(),"cE07000075");
data.put("Tendring".toUpperCase(),"cE07000076");
data.put("Uttlesford".toUpperCase(),"cE07000077");
data.put("Cheltenham".toUpperCase(),"cE07000078");
data.put("Cotswold".toUpperCase(),"cE07000079");
data.put("Forest of Dean".toUpperCase(),"cE07000080");
data.put("Gloucester".toUpperCase(),"cE07000081");
data.put("Stroud".toUpperCase(),"cE07000082");
data.put("Tewkesbury".toUpperCase(),"cE07000083");
data.put("Basingstoke and Deane".toUpperCase(),"cE07000084");
data.put("East Hampshire".toUpperCase(),"cE07000085");
data.put("Eastleigh".toUpperCase(),"cE07000086");
data.put("Fareham".toUpperCase(),"cE07000087");
data.put("Gosport".toUpperCase(),"cE07000088");
data.put("Hart".toUpperCase(),"cE07000089");
data.put("Havant".toUpperCase(),"cE07000090");
data.put("New Forest".toUpperCase(),"cE07000091");
data.put("Rushmoor".toUpperCase(),"cE07000092");
data.put("Test Valley".toUpperCase(),"cE07000093");
data.put("Winchester".toUpperCase(),"cE07000094");
data.put("Broxbourne".toUpperCase(),"cE07000095");
data.put("Dacorum".toUpperCase(),"cE07000096");
data.put("Hertsmere".toUpperCase(),"cE07000098");
data.put("North Hertfordshire".toUpperCase(),"cE07000099");
data.put("Three Rivers".toUpperCase(),"cE07000102");
data.put("Watford".toUpperCase(),"cE07000103");
data.put("Ashford".toUpperCase(),"cE07000105");
data.put("Canterbury".toUpperCase(),"cE07000106");
data.put("Dartford".toUpperCase(),"cE07000107");
data.put("Dover".toUpperCase(),"cE07000108");
data.put("Gravesham".toUpperCase(),"cE07000109");
data.put("Maidstone".toUpperCase(),"cE07000110");
data.put("Sevenoaks".toUpperCase(),"cE07000111");
data.put("Shepway".toUpperCase(),"cE07000112");
data.put("Swale".toUpperCase(),"cE07000113");
data.put("Thanet".toUpperCase(),"cE07000114");
data.put("Tonbridge and Malling".toUpperCase(),"cE07000115");
data.put("Tunbridge Wells".toUpperCase(),"cE07000116");
data.put("Burnley".toUpperCase(),"cE07000117");
data.put("Chorley".toUpperCase(),"cE07000118");
data.put("Fylde".toUpperCase(),"cE07000119");
data.put("Hyndburn".toUpperCase(),"cE07000120");
data.put("Lancaster".toUpperCase(),"cE07000121");
data.put("Pendle".toUpperCase(),"cE07000122");
data.put("Preston".toUpperCase(),"cE07000123");
data.put("Ribble Valley".toUpperCase(),"cE07000124");
data.put("Rossendale".toUpperCase(),"cE07000125");
data.put("South Ribble".toUpperCase(),"cE07000126");
data.put("West Lancashire".toUpperCase(),"cE07000127");
data.put("Wyre".toUpperCase(),"cE07000128");
data.put("Blaby".toUpperCase(),"cE07000129");
data.put("Charnwood".toUpperCase(),"cE07000130");
data.put("Harborough".toUpperCase(),"cE07000131");
data.put("Hinckley and Bosworth".toUpperCase(),"cE07000132");
data.put("Melton".toUpperCase(),"cE07000133");
data.put("North West Leicestershire".toUpperCase(),"cE07000134");
data.put("Oadby and Wigston".toUpperCase(),"cE07000135");
data.put("Boston".toUpperCase(),"cE07000136");
data.put("East Lindsey".toUpperCase(),"cE07000137");
```

```
data.put("Lincoln".toUpperCase(),"cE07000138");
data.put("North Kesteven".toUpperCase(),"cE07000139");
data.put("South Holland".toUpperCase(),"cE07000140");
data.put("South Kesteven".toUpperCase(),"cE07000141");
data.put("West Lindsey".toUpperCase(),"cE07000142");
data.put("Breckland".toUpperCase(),"cE07000143");
data.put("Broadland".toUpperCase(),"cE07000144");
data.put("Great Yarmouth".toUpperCase(),"cE07000145");
data.put("King's Lynn and West Norfolk".toUpperCase(),"cE07000146");
data.put("North Norfolk".toUpperCase(),"cE07000147");
data.put("Norwich".toUpperCase(),"cE07000148");
data.put("South Norfolk".toUpperCase(),"cE07000149");
data.put("Corby".toUpperCase(),"cE07000150");
data.put("Daventry".toUpperCase(),"cE07000151");
data.put("East Northamptonshire".toUpperCase(),"cE07000152");
data.put("Kettering".toUpperCase(),"cE07000153");
data.put("Northampton".toUpperCase(),"cE07000154");
data.put("South Northamptonshire".toUpperCase(),"cE07000155");
data.put("Wellingborough".toUpperCase(),"cE07000156");
data.put("Craven".toUpperCase(),"cE07000163");
data.put("Hambleton".toUpperCase(),"cE07000164");
data.put("Harrogate".toUpperCase(),"cE07000165");
data.put("Richmondshire".toUpperCase(),"cE07000166");
data.put("Ryedale".toUpperCase(),"cE07000167");
data.put("Scarborough".toUpperCase(),"cE07000168");
data.put("Selby".toUpperCase(),"cE07000169");
data.put("Ashfield".toUpperCase(),"cE07000170");
data.put("Bassetlaw".toUpperCase(),"cE07000171");
data.put("Broxtowe".toUpperCase(),"cE07000172");
data.put("Gedling".toUpperCase(),"cE07000173");
data.put("Mansfield".toUpperCase(),"cE07000174");
data.put("Newark and Sherwood".toUpperCase(),"cE07000175");
data.put("Rushcliffe".toUpperCase(),"cE07000176");
data.put("Cherwell".toUpperCase(),"cE07000177");
data.put("Oxford".toUpperCase(),"cE07000178");
data.put("South Oxfordshire".toUpperCase(),"cE07000179");
data.put("Vale of White Horse".toUpperCase(),"cE07000180");
data.put("West Oxfordshire".toUpperCase(),"cE07000181");
data.put("Mendip".toUpperCase(),"cE07000187");
data.put("Sedgemoor".toUpperCase(),"cE07000188");
data.put("South Somerset".toUpperCase(),"cE07000189");
data.put("Taunton Deane".toUpperCase(),"cE07000190");
data.put("West Somerset".toUpperCase(),"cE07000191");
data.put("Cannock Chase".toUpperCase(),"cE07000192");
data.put("East Staffordshire".toUpperCase(),"cE07000193");
data.put("Lichfield".toUpperCase(),"cE07000194");
data.put("Newcastle-under-Lyme".toUpperCase(),"cE07000195");
data.put("South Staffordshire".toUpperCase(),"cE07000196");
data.put("Stafford".toUpperCase(),"cE07000197");
data.put("Staffordshire Moorlands".toUpperCase(),"cE07000198");
data.put("Tamworth".toUpperCase(),"cE07000199");
data.put("Babergh".toUpperCase(),"cE07000200");
data.put("Forest Heath".toUpperCase(),"cE07000201");
data.put("Ipswich".toUpperCase(),"cE07000202");
data.put("Mid Suffolk".toUpperCase(),"cE07000203");
data.put("St Edmundsbury".toUpperCase(),"cE07000204");
data.put("Suffolk Coastal".toUpperCase(),"cE07000205");
data.put("Waveney".toUpperCase(),"cE07000206");
data.put("Elmbridge".toUpperCase(),"cE07000207");
data.put("Epsom and Ewell".toUpperCase(),"cE07000208");
data.put("Guildford".toUpperCase(),"cE07000209");
data.put("Mole Valley".toUpperCase(),"cE07000210");
data.put("Reigate and Banstead".toUpperCase(),"cE07000211");
data.put("Runnymede".toUpperCase(),"cE07000212");
data.put("Spelthorne".toUpperCase(),"cE07000213");
```

```
data.put("Surrey Heath".toUpperCase(),"cE07000214");
data.put("Tandridge".toUpperCase(),"cE07000215");
data.put("Waverley".toUpperCase(),"cE07000216");
data.put("Woking".toUpperCase(),"cE07000217");
data.put("North Warwickshire".toUpperCase(),"cE07000218");
data.put("Nuneaton and Bedworth".toUpperCase(),"cE07000219");
data.put("Rugby".toUpperCase(),"cE07000220");
data.put("Stratford-on-Avon".toUpperCase(),"cE07000221");
data.put("Warwick".toUpperCase(),"cE07000222");
data.put("Adur".toUpperCase(),"cE07000223");
data.put("Arun".toUpperCase(),"cE07000224");
data.put("Chichester".toUpperCase(),"cE07000225");
data.put("Crawley".toUpperCase(),"cE07000226");
data.put("Horsham".toUpperCase(),"cE07000227");
data.put("Mid Sussex".toUpperCase(),"cE07000228");
data.put("Worthing".toUpperCase(),"cE07000229");
data.put("Bromsgrove".toUpperCase(),"cE07000234");
data.put("Malvern Hills".toUpperCase(),"cE07000235");
data.put("Redditch".toUpperCase(),"cE07000236");
data.put("Worcester".toUpperCase(),"cE07000237");
data.put("Wychavon".toUpperCase(),"cE07000238");
data.put("Wyre Forest".toUpperCase(),"cE07000239");
data.put("St Albans".toUpperCase(),"cE07000240");
data.put("Welwyn Hatfield".toUpperCase(),"cE07000241");
data.put("East Hertfordshire".toUpperCase(),"cE07000242");
data.put("Stevenage".toUpperCase(),"cE07000243");
data.put("Bolton".toUpperCase(),"cE08000001");
data.put("Bury".toUpperCase(),"cE08000002");
data.put("Manchester".toUpperCase(),"cE08000003");
data.put("Oldham".toUpperCase(),"cE08000004");
data.put("Rochdale".toUpperCase(),"cE08000005");
data.put("Salford".toUpperCase(),"cE08000006");
data.put("Stockport".toUpperCase(),"cE08000007");
data.put("Tameside".toUpperCase(),"cE08000008");
data.put("Trafford".toUpperCase(),"cE08000009");
data.put("Wigan".toUpperCase(),"cE08000010");
data.put("Knowsley".toUpperCase(),"cE08000011");
data.put("Liverpool".toUpperCase(),"cE08000012");
data.put("St. Helens".toUpperCase(),"cE08000013");
data.put("Sefton".toUpperCase(),"cE08000014");
data.put("Wirral".toUpperCase(),"cE08000015");
data.put("Barnsley".toUpperCase(),"cE08000016");
data.put("Doncaster".toUpperCase(),"cE08000017");
data.put("Rotherham".toUpperCase(),"cE08000018");
data.put("Sheffield".toUpperCase(),"cE08000019");
data.put("Newcastle upon Tyne".toUpperCase(),"cE08000021");
data.put("North Tyneside".toUpperCase(),"cE08000022");
data.put("South Tyneside".toUpperCase(),"cE08000023");
data.put("Sunderland".toUpperCase(),"cE08000024");
data.put("Birmingham".toUpperCase(),"cE08000025");
data.put("Coventry".toUpperCase(),"cE08000026");
data.put("Dudley".toUpperCase(),"cE08000027");
data.put("Sandwell".toUpperCase(),"cE08000028");
data.put("Solihull".toUpperCase(),"cE08000029");
data.put("Walsall".toUpperCase(),"cE08000030");
data.put("Wolverhampton".toUpperCase(),"cE08000031");
data.put("Bradford".toUpperCase(),"cE08000032");
data.put("Calderdale".toUpperCase(),"cE08000033");
data.put("Kirklees".toUpperCase(),"cE08000034");
data.put("Leeds".toUpperCase(),"cE08000035");
data.put("Wakefield".toUpperCase(),"cE08000036");
data.put("Gateshead".toUpperCase(),"cE08000037");
data.put("City of London".toUpperCase(),"cE09000001");
data.put("Barking and Dagenham".toUpperCase(),"cE09000002");
data.put("Barnet".toUpperCase(),"cE09000003");
```

```
data.put("Bexley".toUpperCase(),"cE09000004");
data.put("Brent".toUpperCase(),"cE09000005");
data.put("Bromley".toUpperCase(),"cE09000006");
data.put("Camden".toUpperCase(),"cE09000007");
data.put("Croydon".toUpperCase(),"cE09000008");
data.put("Ealing".toUpperCase(),"cE09000009");
data.put("Enfield".toUpperCase(),"cE09000010");
data.put("Greenwich".toUpperCase(),"cE09000011");
data.put("Hackney".toUpperCase(),"cE09000012");
data.put("Hammersmith and Fulham".toUpperCase(),"cE09000013");
data.put("Haringey".toUpperCase(),"cE09000014");
data.put("Harrow".toUpperCase(),"cE09000015");
data.put("Havering".toUpperCase(),"cE09000016");
data.put("Hillingdon".toUpperCase(),"cE09000017");
data.put("Hounslow".toUpperCase(),"cE09000018");
data.put("Islington".toUpperCase(),"cE09000019");
data.put("Kensington and Chelsea".toUpperCase(),"cE09000020");
data.put("Kingston upon Thames".toUpperCase(),"cE09000021");
data.put("Lambeth".toUpperCase(),"cE09000022");
data.put("Lewisham".toUpperCase(),"cE09000023");
data.put("Merton".toUpperCase(),"cE09000024");
data.put("Newham".toUpperCase(),"cE09000025");
data.put("Redbridge".toUpperCase(),"cE09000026");
data.put("Richmond upon Thames".toUpperCase(),"cE09000027");
data.put("Southwark".toUpperCase(),"cE09000028");
data.put("Sutton".toUpperCase(),"cE09000029");
data.put("Tower Hamlets".toUpperCase(),"cE09000030");
data.put("Waltham Forest".toUpperCase(),"cE09000031");
data.put("Wandsworth".toUpperCase(),"cE09000032");
data.put("Westminster".toUpperCase(),"cE09000033");
data.put("Clackmannanshire".toUpperCase(),"cS12000005");
data.put("Dumfries and Galloway".toUpperCase(),"cS12000006");
data.put("East Ayrshire".toUpperCase(),"cS12000008");
data.put("East Lothian".toUpperCase(),"cS12000010");
data.put("East Renfrewshire".toUpperCase(),"cS12000011");
data.put("Eilean Siar".toUpperCase(),"cS12000013");
data.put("Falkirk".toUpperCase(),"cS12000014");
data.put("Fife".toUpperCase(),"cS12000015");
data.put("Highland".toUpperCase(),"cS12000017");
data.put("Inverclyde".toUpperCase(),"cS12000018");
data.put("Midlothian".toUpperCase(),"cS12000019");
data.put("Moray".toUpperCase(),"cS12000020");
data.put("North Ayrshire".toUpperCase(),"cS12000021");
data.put("Orkney Islands".toUpperCase(),"cS12000023");
data.put("Perth and Kinross".toUpperCase(),"cS12000024");
data.put("Scottish Borders".toUpperCase(),"cS12000026");
data.put("Shetland Islands".toUpperCase(),"cS12000027");
data.put("South Ayrshire".toUpperCase(),"cS12000028");
data.put("South Lanarkshire".toUpperCase(),"cS12000029");
data.put("Stirling".toUpperCase(),"cS12000030");
data.put("Aberdeen City".toUpperCase(),"cS12000033");
data.put("Aberdeenshire".toUpperCase(),"cS12000034");
data.put("Argyll and Bute".toUpperCase(),"cS12000035");
data.put("City of Edinburgh".toUpperCase(),"cS12000036");
data.put("Renfrewshire".toUpperCase(),"cS12000038");
data.put("West Dunbartonshire".toUpperCase(),"cS12000039");
data.put("West Lothian".toUpperCase(),"cS12000040");
data.put("Angus".toUpperCase(),"cS12000041");
data.put("Dundee City".toUpperCase(),"cS12000042");
data.put("North Lanarkshire".toUpperCase(),"cS12000044");
data.put("East Dunbartonshire".toUpperCase(),"cS12000045");
data.put("Glasgow City".toUpperCase(),"cS12000046");
data.put("Isle of Anglesey".toUpperCase(),"cW06000001");
data.put("Gwynedd".toUpperCase(),"cW06000002");
data.put("Conwy".toUpperCase(),"cW06000003");
```

```
                data.put("Denbighshire".toUpperCase(),"cW06000004");
                data.put("Flintshire".toUpperCase(),"cW06000005");
                data.put("Wrexham".toUpperCase(),"cW06000006");
                data.put("Ceredigion".toUpperCase(),"cW06000008");
                data.put("Pembrokeshire".toUpperCase(),"cW06000009");
                data.put("Carmarthenshire".toUpperCase(),"cW06000010");
                data.put("Swansea".toUpperCase(),"cW06000011");
                data.put("Neath Port Talbot".toUpperCase(),"cW06000012");
                data.put("Bridgend".toUpperCase(),"cW06000013");
                data.put("Vale of Glamorgan".toUpperCase(),"cW06000014");
                data.put("Cardiff".toUpperCase(),"cW06000015");
                data.put("Rhondda Cynon Taf".toUpperCase(),"cW06000016");
                data.put("Caerphilly".toUpperCase(),"cW06000018");
                data.put("Blaenau Gwent".toUpperCase(),"cW06000019");
                data.put("Torfaen".toUpperCase(),"cW06000020");
                data.put("Monmouthshire".toUpperCase(),"cW06000021");
                data.put("Newport".toUpperCase(),"cW06000022");
                data.put("Powys".toUpperCase(),"cW06000023");
                data.put("Merthyr Tydfil".toUpperCase(),"cW06000024");

        }
}
```

1.6.2 Track Keywords (TrackKeywords.java)

```
package twitter.storm.tools;

import org.ahocorasick.trie.Trie;

/**
 * Class that initializes the Trie library
 * to track the flu words in tweets
 */

public class TrackKeywords {

        Trie searchTrie;

        public Trie getSearchTrie() {
                return searchTrie;
        }

        public TrackKeywords() {

                searchTrie = new Trie().onlyWholeWords().caseInsensitive();

            searchTrie.addKeyword("vomit");
            searchTrie.addKeyword("sorethroat");
            searchTrie.addKeyword("headache");
            searchTrie.addKeyword("aches");
            searchTrie.addKeyword("feel sick");
            searchTrie.addKeyword("feeling sick");
            searchTrie.addKeyword("runny nose");
            searchTrie.addKeyword("catarrh");
            searchTrie.addKeyword("feeling unwell");
            searchTrie.addKeyword("shivers");
            searchTrie.addKeyword("fever");
            searchTrie.addKeyword("sick");
            searchTrie.addKeyword("#flu");
```

```
            searchTrie.addKeyword("#fluseason");
            searchTrie.addKeyword("flu");
            searchTrie.addKeyword("sneezes");
            searchTrie.addKeyword("gastroenteritis");
            searchTrie.addKeyword("sore throat");
            searchTrie.addKeyword("cough");
            searchTrie.addKeyword("spasm");
            searchTrie.addKeyword("stomach ache");
            searchTrie.addKeyword("inflammation");
            searchTrie.addKeyword("diarrhea");
            searchTrie.addKeyword("muscles ache");
            searchTrie.addKeyword("muscles pain");
            searchTrie.addKeyword("nausea");
             searchTrie.addKeyword("exhausted");
             searchTrie.addKeyword("exhaustion");

        }

}
```

1.6.3 Cities Lookup (CitiesLookup.java)

```
package twitter.storm.tools;

import java.util.HashMap;

/**
 * Class that initializes a HashMap
 * of major UK city names as keys and
 * their coordinates as values
 */
public class CitiesLookup {

        // Holds all major uk cities with their coordinates
        private HashMap<String, String> cities;

        public HashMap<String, String> getCities() {
                return cities;
        }

        public CitiesLookup() {

                cities = new HashMap<String, String>();

                cities.put("Aberaeron".toUpperCase(),"52.242793,-4.258161900000005");
                cities.put("Aberdare".toUpperCase(),"51.716154,-3.451816000000008");
                cities.put("Aberdeen".toUpperCase(),"57.149717,-2.094278000000031");
                cities.put("Aberfeldy".toUpperCase(),"56.621752,-3.86696900000004");
                cities.put("Abergavenny".toUpperCase(),"51.825366,-3.0194229999999607");
                cities.put("Abergele".toUpperCase(),"53.284355,-3.581405000000018");
                cities.put("Abertillery".toUpperCase(),"51.732161,-3.1353689999999688");
                cities.put("Aberystwyth".toUpperCase(),"52.41530299999999,-4.082920000000058");
                cities.put("Abingdon".toUpperCase(),"51.67078,-1.2879528999999366");
                cities.put("Accrington".toUpperCase(),"53.753609,-2.3721800000000712");
                cities.put("Adlington".toUpperCase(),"53.611055,-2.6063649999999825");
                cities.put("Airdrie".toUpperCase(),"55.866267,-3.962566000000038");
                cities.put("Alcester".toUpperCase(),"52.215311,-1.867605000000026");
                cities.put("Aldeburgh".toUpperCase(),"52.155357,1.6004460000000336");
                cities.put("Aldershot".toUpperCase(),"51.248366,-0.7557509000000664");
```

```
                cities.put("Aldridge".toUpperCase(),"52.6110324,-1.9142056000000593");
                cities.put("Alford".toUpperCase(),"53.263271,0.18335899999999583");
                cities.put("Alfreton".toUpperCase(),"53.09744999999999,-1.382255999999984");
                cities.put("Alloa".toUpperCase(),"56.114073,-3.7918959999999515");
                cities.put("Alnwick".toUpperCase(),"55.412744,-1.7062989999999445");
                cities.put("Alsager".toUpperCase(),"53.095402,-2.30465700000002");
                cities.put("Alston".toUpperCase(),"54.812167,-2.438684999999964");
                cities.put("Amesbury".toUpperCase(),"51.1679201,-1.7629782999999861");
                cities.put("Amlwch".toUpperCase(),"53.410658,-4.345692999999983");
                cities.put("Ammanford".toUpperCase(),"51.792888,-3.9884590000000344");
                cities.put("Ampthill".toUpperCase(),"52.0272503,-0.4951416999999765");
                cities.put("Andover".toUpperCase(),"51.2111975,-1.4919233000000531");
                cities.put("Annan".toUpperCase(),"54.990246,-3.2597729999999956");
                cities.put("Antrim".toUpperCase(),"54.7195338,-6.2072498");
                cities.put("Appleby in Westmorland".toUpperCase(),"54.578302,-
2.4896139999999605");
                cities.put("Arbroath".toUpperCase(),"56.559107,-2.5915430000000015");
                cities.put("Armagh".toUpperCase(),"54.3502798,-6.652791999999977");
                cities.put("Arundel".toUpperCase(),"50.855171,-0.5551189999999906");
                cities.put("Ashbourne".toUpperCase(),"53.016626,-1.7321789999999737");
                cities.put("Ashburton".toUpperCase(),"50.515772,-3.7513950000000023");
                cities.put("Ashby de la Zouch".toUpperCase(),"52.749212,-1.4686179999999922");
                cities.put("Ashford".toUpperCase(),"51.1464659,0.8750190000000657");
                cities.put("Ashington".toUpperCase(),"55.182978,-1.5659828999999945");
                cities.put("Ashton in Makerfield".toUpperCase(),"53.4869569,-
2.642565999999988");
                cities.put("Atherstone".toUpperCase(),"52.576614,-1.5437630000000127");
                cities.put("Auchtermuchty".toUpperCase(),"56.292291,-3.236091999999985");
                cities.put("Axminster".toUpperCase(),"50.78272699999999,-2.9949369999999362");
                cities.put("Aylesbury".toUpperCase(),"51.815606,-0.808400000000006");
                cities.put("Aylsham".toUpperCase(),"52.796672,1.2523519999999735");
                cities.put("Ayr".toUpperCase(),"55.458564,-4.629179000000022");
                cities.put("Bacup".toUpperCase(),"53.707167,-2.2012589999999363");
                cities.put("Bakewell".toUpperCase(),"53.215207,-1.6761710000000676");
                cities.put("Bala".toUpperCase(),"52.909911,-3.6015840000000026");
                cities.put("Ballater".toUpperCase(),"57.05003,-3.0366490000000113");
                cities.put("Ballycastle".toUpperCase(),"55.2052888,-6.2532946999999695");
                cities.put("Ballyclare".toUpperCase(),"54.75291439999999,-6.001522099999988");
                cities.put("Ballymena".toUpperCase(),"54.8652935,-6.280221299999994");
                cities.put("Ballymoney".toUpperCase(),"55.0704888,-6.517370799999981");
                cities.put("Ballynahinch".toUpperCase(),"54.4028842,-5.8974313999999595");
                cities.put("Banbridge".toUpperCase(),"54.348729,-6.270480300000031");
                cities.put("Banbury".toUpperCase(),"52.0629009,-1.3397750000000315");
                cities.put("Banchory".toUpperCase(),"57.053856,-2.4909599999999728");
                cities.put("Banff".toUpperCase(),"57.666505,-2.5240380000000187");
                cities.put("Bangor".toUpperCase(),"53.22739,-4.129263000000037");
                cities.put("Barmouth".toUpperCase(),"52.722873,-4.056091899999956");
                cities.put("Barnard Castle".toUpperCase(),"54.545284,-1.9237410000000637");
                cities.put("Barnet".toUpperCase(),"51.6569225,-0.1949251999999433");
                cities.put("Barnoldswick".toUpperCase(),"53.91301499999999,-
2.1869179999999915");
                cities.put("Barnsley".toUpperCase(),"53.55263,-1.4797260000000279");
                cities.put("Barnstaple".toUpperCase(),"51.0781599,-4.058338000000049");
                cities.put("Barrhead".toUpperCase(),"55.79671099999999,-4.397713999999951");
                cities.put("Barrow in Furness".toUpperCase(),"54.108967,-3.2188939999999775");
                cities.put("Barry".toUpperCase(),"56.9808838,-7.456795899999975");
                cities.put("Barton upon Humber".toUpperCase(),"53.686675,-0.443367999999964");
                cities.put("Basildon".toUpperCase(),"51.57608399999999,0.48873600000001716");
                cities.put("Basingstoke".toUpperCase(),"51.26654,-1.0923963999999842");
                cities.put("Bath".toUpperCase(),"51.375801,-2.359903900000063");
                cities.put("Bathgate".toUpperCase(),"55.9024,-3.6431179999999586");
                cities.put("Batley".toUpperCase(),"53.717028,-1.6350830000000087");
                cities.put("Battle".toUpperCase(),"50.917405,0.48367899999993824");
                cities.put("Bawtry".toUpperCase(),"53.43125,-1.0185149999999794");
                cities.put("Beaconsfield".toUpperCase(),"51.602396,-0.6442408999999998");
```

```
cities.put("Bearsden".toUpperCase(),"55.92169,-4.335524999999961");
cities.put("Beaumaris".toUpperCase(),"53.265865,-4.09199000000001");
cities.put("Bebington".toUpperCase(),"53.3530146,-3.0075315000000273");
cities.put("Beccles".toUpperCase(),"52.459333,1.5660530000000108");
cities.put("Bedale".toUpperCase(),"54.2887008,-1.5933178999999882");
cities.put("Bedford".toUpperCase(),"52.1359729,-0.46665459999996983");
cities.put("Bedlington".toUpperCase(),"55.131664,-1.599454000000037");
cities.put("Bedworth".toUpperCase(),"52.481392,-1.4688690000000406");
cities.put("Beeston".toUpperCase(),"52.92392,-1.2123940000000175");
cities.put("Bellshill".toUpperCase(),"55.816761,-4.026535999999965");
cities.put("Belper".toUpperCase(),"53.0243899,-1.4776160000000118");
cities.put("Berkhamsted".toUpperCase(),"51.75976600000001,-0.567856900000038");
cities.put("Berwick upon Tweed".toUpperCase(),"55.770242,-2.0053950000000214");
cities.put("Betws y Coed".toUpperCase(),"53.0930858,-3.801035400000046");
cities.put("Beverley".toUpperCase(),"53.841963,-0.4350930000000517");
cities.put("Bewdley".toUpperCase(),"52.375539,-2.3169729999999618");
cities.put("Bexhill on Sea".toUpperCase(),"50.8499062,0.46620710000001964");
cities.put("Bicester".toUpperCase(),"51.899603,-1.1535899000000427");
cities.put("Biddulph".toUpperCase(),"53.119197,-2.171607999999992");
cities.put("Bideford".toUpperCase(),"51.016684,-4.206666000000041");
cities.put("Biggar".toUpperCase(),"55.623396,-3.523963999999978");
cities.put("Biggleswade".toUpperCase(),"52.086938,-0.264220000000023");
cities.put("Billericay".toUpperCase(),"51.627903,0.4183970000000272");
cities.put("Bilston".toUpperCase(),"52.56559499999999,-2.0740879999999606");
cities.put("Bingham".toUpperCase(),"52.951088,-0.9561439999999948");
cities.put("Birkenhead".toUpperCase(),"53.38999099999999,-3.023009000000002");
cities.put("Birmingham".toUpperCase(),"52.48624299999999,-1.8904009999999971");
cities.put("Bishop Auckland".toUpperCase(),"54.663822,-1.6787879999999404");
cities.put("Blackburn".toUpperCase(),"53.748575,-2.487528999999995");
cities.put("Blackheath".toUpperCase(),"51.206182,-0.5277690000000348");
cities.put("Blackpool".toUpperCase(),"53.8175053,-3.035674800000038");
cities.put("Blaenau Ffestiniog".toUpperCase(),"52.998337,-3.9448939999999766");
cities.put("Blandford Forum".toUpperCase(),"50.856908,-2.165352999999982");
cities.put("Bletchley".toUpperCase(),"52.002175,-0.7501369999999952");
cities.put("Bloxwich".toUpperCase(),"52.6165017,-1.99760660000004");
cities.put("Blyth".toUpperCase(),"55.126957,-1.5102769999999737");
cities.put("Bodmin".toUpperCase(),"50.46863,-4.715113999999971");
cities.put("Bognor Regis".toUpperCase(),"50.782998,-0.6730609999999615");
cities.put("Bollington".toUpperCase(),"53.29669999999999,-2.096651999999949");
cities.put("Bolsover".toUpperCase(),"53.231044,-1.2897209000000203");
cities.put("Bolton".toUpperCase(),"53.57686469999999,-2.4282192000000578");
cities.put("Bootle".toUpperCase(),"53.44325500000001,-2.9988949999999477");
cities.put("Borehamwood".toUpperCase(),"51.657728,-0.2723080000000664");
cities.put("Boston".toUpperCase(),"52.97893999999999,-0.02657699999997476");
cities.put("Bourne".toUpperCase(),"52.76859899999999,-0.3784620000000132");
cities.put("Bournemouth".toUpperCase(),"50.719164,-1.8807689999999866");
cities.put("Brackley".toUpperCase(),"52.027411,-1.1431519999999864");
cities.put("Bracknell".toUpperCase(),"51.41604,-0.7539799999999559");
cities.put("Bradford on Avon".toUpperCase(),"51.345178,-2.25250189999997");
cities.put("Bradford".toUpperCase(),"53.795984,-1.7593980000000329");
cities.put("Brading".toUpperCase(),"50.679825,-1.1445399000000407");
cities.put("Bradley Stoke".toUpperCase(),"51.532291,-2.5462119999999686");
cities.put("Bradninch".toUpperCase(),"50.828878,-3.4182100000000446");
cities.put("Braintree".toUpperCase(),"51.880087,0.5509269000000359");
cities.put("Brechin".toUpperCase(),"56.73334200000001,-2.6552888999999595");
cities.put("Brecon".toUpperCase(),"51.9489469,-3.3914629999999306");
cities.put("Brentwood".toUpperCase(),"51.620475,0.30717490000006364");
cities.put("Bridge of Allan".toUpperCase(),"56.153149,-3.942207999999937");
cities.put("Bridgend".toUpperCase(),"51.504286,-3.5769450000000234");
cities.put("Bridgnorth".toUpperCase(),"52.53445499999999,-2.42454900000007");
cities.put("Bridgwater".toUpperCase(),"51.127889,-3.0036320000000387");
cities.put("Bridlington".toUpperCase(),"54.08535,-0.1988020000000006");
cities.put("Bridport".toUpperCase(),"50.7335769,-2.7583009999999604");
cities.put("Brigg".toUpperCase(),"53.55184999999999,-0.4918870000000197");
cities.put("Brighouse".toUpperCase(),"53.699729,-1.7825010000000248");
```

```
cities.put("Brightlingsea".toUpperCase(),"51.816142,1.021398999999974");
cities.put("Brighton".toUpperCase(),"50.82253000000001,-0.13716299999998682");
cities.put("Bristol".toUpperCase(),"51.454513,-2.5879099999999653");
cities.put("Brixham".toUpperCase(),"50.39514,-3.5139239999999745");
cities.put("Broadstairs".toUpperCase(),"51.360163,1.4320380000000341");
cities.put("Bromsgrove".toUpperCase(),"52.335589,-2.0619060000000218");
cities.put("Bromyard".toUpperCase(),"52.190713,-2.5087200000000394");
cities.put("Brynmawr".toUpperCase(),"51.795089,-3.175561000000016");
cities.put("Buckfastleigh".toUpperCase(),"50.481799,-3.7793420000000424");
cities.put("Buckie".toUpperCase(),"57.67739199999999,-2.967310999999995");
cities.put("Buckingham".toUpperCase(),"51.999326,-0.9876449999999295");
cities.put("Buckley".toUpperCase(),"53.17005,-3.0814878999999564");
cities.put("Bude".toUpperCase(),"50.826636,-4.543678");
cities.put("Budleigh Salterton".toUpperCase(),"50.630998,-3.3202139999999645");
cities.put("Builth Wells".toUpperCase(),"52.150023,-3.4045919999999796");
cities.put("Bungay".toUpperCase(),"52.455457,1.4439760000000206");
cities.put("Buntingford".toUpperCase(),"51.946217,-0.018547000000012304");
cities.put("Burford".toUpperCase(),"51.807083,-1.636790000000019");
cities.put("Burgess Hill".toUpperCase(),"50.954469,-0.1287009999999782");
cities.put("Burnham on Crouch".toUpperCase(),"51.628347,0.8145389999999679");
cities.put("Burnham on Sea".toUpperCase(),"51.240139,-2.9938580000000456");
cities.put("Burnley".toUpperCase(),"53.7892877,-2.2405034999999316");
cities.put("Burntisland".toUpperCase(),"56.058809,-3.2339150000000245");
cities.put("Burntwood".toUpperCase(),"52.679936,-1.92178100000001");
cities.put("Burry Port".toUpperCase(),"51.683466,-4.25610000000006");
cities.put("Burton Latimer".toUpperCase(),"52.365906,-0.6782080000000406");
cities.put("Bury".toUpperCase(),"53.5933498,-2.296605399999976");
cities.put("Bushmills".toUpperCase(),"55.20506,-6.52346");
cities.put("Buxton".toUpperCase(),"53.259082,-1.914829999999938");
cities.put("Caernarfon".toUpperCase(),"53.139551,-4.273910999999998");
cities.put("Caerphilly".toUpperCase(),"51.578829,-3.2181339999999636");
cities.put("Caistor".toUpperCase(),"53.497485,-0.31539299999997183");
cities.put("Caldicot".toUpperCase(),"51.591518,-2.7508679999999686");
cities.put("Callander".toUpperCase(),"56.245059,-4.211663000000044");
cities.put("Calne".toUpperCase(),"51.43933,-2.003883999999971");
cities.put("Camberley".toUpperCase(),"51.3353899,-0.7428559999999607");
cities.put("Camborne".toUpperCase(),"50.21277,-5.294774899999993");
cities.put("Cambridge".toUpperCase(),"52.205337,0.12181699999996454");
cities.put("Camelford".toUpperCase(),"50.622055,-4.682364000000007");
cities.put("Campbeltown".toUpperCase(),"55.424117,-5.605373999999983");
cities.put("Cannock".toUpperCase(),"52.69994080000001,-2.021829300000036");
cities.put("Canterbury".toUpperCase(),"51.280233,1.0789088999999876");
cities.put("Cardiff".toUpperCase(),"51.48158100000001,-3.1790899999999738");
cities.put("Cardigan".toUpperCase(),"52.083703,-4.660863999999947");
cities.put("Carlisle".toUpperCase(),"54.892473,-2.9329310000000532");
cities.put("Carluke".toUpperCase(),"55.735435,-3.8365519999999833");
cities.put("Carmarthen".toUpperCase(),"51.85762,-4.312131000000022");
cities.put("Carnforth".toUpperCase(),"54.127363,-2.7681119999999737");
cities.put("Carnoustie".toUpperCase(),"56.502565,-2.7032500000000255");
cities.put("Carrickfergus".toUpperCase(),"54.7261871,-5.81012069999997");
cities.put("Carterton".toUpperCase(),"51.759712,-1.5931239999999889");
cities.put("Castle Douglas".toUpperCase(),"54.940415,-3.931414000000018");
cities.put("Castlederg".toUpperCase(),"54.7097899,-7.593671399999948");
cities.put("Castleford".toUpperCase(),"53.72346599999999,-1.3459679999999707");
cities.put("Castlewellan".toUpperCase(),"54.2579967,-5.941821900000036");
cities.put("Chard".toUpperCase(),"50.8697919,-2.9633159999999634");
cities.put("Charlbury".toUpperCase(),"51.8745,-1.4797519999999622");
cities.put("Chatham".toUpperCase(),"51.380952,0.522130000000061");
cities.put("Chatteris".toUpperCase(),"52.45610199999999,0.054012000000057014");
cities.put("Chelmsford".toUpperCase(),"51.7355868,0.4685497000000396");
cities.put("Cheltenham".toUpperCase(),"51.8993855,-2.0782533000000285");
cities.put("Chepstow".toUpperCase(),"51.641856,-2.6738040000000183");
cities.put("Chesham".toUpperCase(),"51.709401,-0.6123330000000351");
cities.put("Cheshunt".toUpperCase(),"51.69988799999999,-0.028485999999929845");
cities.put("Chester le Street".toUpperCase(),"54.8591161,-1.5740888000000268");
```

```
                cities.put("Chester".toUpperCase(),"53.193392,-2.8930749999999534");
                cities.put("Chesterfield".toUpperCase(),"53.235048,-1.421628999999939");
                cities.put("Chichester".toUpperCase(),"50.83761000000001,-0.7749360000000252");
                cities.put("Chippenham".toUpperCase(),"51.461514,-2.1195156999999654");
                cities.put("Chipping Campden".toUpperCase(),"52.04965499999999,-
1.7832210999999916");
                cities.put("Chipping Norton".toUpperCase(),"51.943544,-1.5421890000000076");
                cities.put("Chipping Sodbury".toUpperCase(),"51.5351659,-2.39093790000004");
                cities.put("Chorley".toUpperCase(),"53.653511,-2.632596000000035");
                cities.put("Christchurch".toUpperCase(),"50.735777,-1.7785860000000184");
                cities.put("Church Stretton".toUpperCase(),"52.540698,-2.803484000000026");
                cities.put("Cinderford".toUpperCase(),"51.825303,-2.500957900000003");
                cities.put("Cirencester".toUpperCase(),"51.718495,-1.9682430000000295");
                cities.put("Clacton on Sea".toUpperCase(),"51.789534,1.1530350000000453");
                cities.put("Cleckheaton".toUpperCase(),"53.723048,-1.7075250000000324");
                cities.put("Cleethorpes".toUpperCase(),"53.557378,-0.02943500000003496");
                cities.put("Clevedon".toUpperCase(),"51.442108,-2.8561959999999544");
                cities.put("Clitheroe".toUpperCase(),"53.871098,-2.3930829999999332");
                cities.put("Clogher".toUpperCase(),"54.4143,-7.164520000000039");
                cities.put("Clydebank".toUpperCase(),"55.900099,-4.404773999999975");
                cities.put("Coalisland".toUpperCase(),"54.5403238,-6.702247499999999");
                cities.put("Coalville".toUpperCase(),"52.724569,-1.36771090000002");
                cities.put("Coatbridge".toUpperCase(),"55.862241,-4.01933699999995");
                cities.put("Cockermouth".toUpperCase(),"54.663261,-3.367984999999976");
                cities.put("Coggeshall".toUpperCase(),"51.870799,0.6926809999999932");
                cities.put("Colchester".toUpperCase(),"51.895927,0.8918740000000298");
                cities.put("Coldstream".toUpperCase(),"55.65132000000001,-2.253398999999945");
                cities.put("Coleraine".toUpperCase(),"55.1325802,-6.66461019999997");
                cities.put("Coleshill".toUpperCase(),"52.499599,-1.706495000000018");
                cities.put("Colne".toUpperCase(),"53.85564,-2.1769910000000436");
                cities.put("Colwyn Bay".toUpperCase(),"53.29318199999999,-3.727640000000065");
                cities.put("Comber".toUpperCase(),"54.5504547,-5.745675099999971");
                cities.put("Congleton".toUpperCase(),"53.162856,-2.218923000000018");
                cities.put("Conwy".toUpperCase(),"53.282872,-3.8294799999999896");
                cities.put("Cookstown".toUpperCase(),"54.6418158,-6.7443894000000455");
                cities.put("Corbridge".toUpperCase(),"54.97404599999999,-2.0175050000000283");
                cities.put("Corby".toUpperCase(),"52.49229829999999,-0.6842332999999599");
                cities.put("Coventry".toUpperCase(),"52.406822,-1.519692999999961");
                cities.put("Cowbridge".toUpperCase(),"51.46209,-3.4495809999999665");
                cities.put("Cowdenbeath".toUpperCase(),"56.114407,-3.3418480000000272");
                cities.put("Cowes".toUpperCase(),"50.76278500000001,-1.300532999999973");
                cities.put("Craigavon".toUpperCase(),"54.4429264,-6.417565299999978");
                cities.put("Cramlington".toUpperCase(),"55.086136,-1.5807999999999538");
                cities.put("Crawley".toUpperCase(),"51.1091401,-0.18722750000006272");
                cities.put("Crayford".toUpperCase(),"51.453003,0.1790969999999561");
                cities.put("Crediton".toUpperCase(),"50.792592,-3.651495000000068");
                cities.put("Crewe".toUpperCase(),"53.10040499999999,-2.4438208999999915");
                cities.put("Crewkerne".toUpperCase(),"50.8830079,-2.7958770000000186");
                cities.put("Criccieth".toUpperCase(),"52.919565,-4.233645000000024");
                cities.put("Crickhowell".toUpperCase(),"51.856479,-3.1355100000000675");
                cities.put("Crieff".toUpperCase(),"56.37654999999999,-3.8419939999999997");
                cities.put("Cromarty".toUpperCase(),"57.680609,-4.034677999999985");
                cities.put("Cromer".toUpperCase(),"52.931448,1.3018660000000182");

cities.put("Crowborough".toUpperCase(),"51.06085299999999,0.16589399999998022");
                cities.put("Crowthorne".toUpperCase(),"51.36690100000001,-0.7953529999999773");
                cities.put("Crumlin".toUpperCase(),"54.6231751,-6.21410190000006");
                cities.put("Cuckfield".toUpperCase(),"51.005813,-0.14363600000001497");
                cities.put("Cullen".toUpperCase(),"57.691635,-2.8219369999999344");
                cities.put("Cullompton".toUpperCase(),"50.85522899999999,-3.391701000000012");
                cities.put("Cumbernauld".toUpperCase(),"55.945668,-3.992533999999978");
                cities.put("Cupar".toUpperCase(),"56.320235,-3.010136999999986");
                cities.put("Cwmbran".toUpperCase(),"51.6496546,-3.0317889000000378");
                cities.put("Dalbeattie".toUpperCase(),"54.934219,-3.8225270000000364");
                cities.put("Dalkeith".toUpperCase(),"55.89311799999999,-3.066605999999979");
```

```
                cities.put("Darlington".toUpperCase(),"54.52361,-1.5594579999999496");
                cities.put("Dartford".toUpperCase(),"51.44621,0.21687199999996665");
                cities.put("Dartmouth".toUpperCase(),"50.352517,-3.5788069999999834");
                cities.put("Darwen".toUpperCase(),"53.695451,-2.4687400000000252");
                cities.put("Daventry".toUpperCase(),"52.257473,-1.1649469999999837");
                cities.put("Dawlish".toUpperCase(),"50.582285,-3.4644020000000637");
                cities.put("Deal".toUpperCase(),"51.222491,1.40166099999999");
                cities.put("Denbigh".toUpperCase(),"53.183906,-3.4250019999999495");
                cities.put("Denton".toUpperCase(),"53.455203,-2.114612999999963");
                cities.put("Derby".toUpperCase(),"52.9225301,-1.4746185999999852");
                cities.put("Dereham".toUpperCase(),"52.68161800000001,0.9378269999999702");
                cities.put("Devizes".toUpperCase(),"51.3489069,-1.994827999999984");
                cities.put("Dewsbury".toUpperCase(),"53.689833,-1.629694900000004");
                cities.put("Didcot".toUpperCase(),"51.60804400000001,-1.2448400000000674");
                cities.put("Dingwall".toUpperCase(),"57.595347,-4.428410999999983");
                cities.put("Dinnington".toUpperCase(),"53.372587,-1.2030680000000302");
                cities.put("Diss".toUpperCase(),"52.37649099999999,1.1083939999999757");
                cities.put("Dolgellau".toUpperCase(),"52.74215,-3.884400000000028");
                cities.put("Donaghadee".toUpperCase(),"54.6425555,-5.537599499999942");
                cities.put("Doncaster".toUpperCase(),"53.52282,-1.1284620000000132");
                cities.put("Dorchester".toUpperCase(),"50.7111639,-2.4411810000000287");
                cities.put("Dorking".toUpperCase(),"51.232202,-0.33237800000006246");
                cities.put("Dornoch".toUpperCase(),"57.87907,-4.028025999999954");
                cities.put("Dover".toUpperCase(),"51.1278758,1.3134026999999833");
                cities.put("Downham Market".toUpperCase(),"52.606612,0.3854719999999361");
                cities.put("Downpatrick".toUpperCase(),"54.32875139999999,-5.715692200000035");
                cities.put("Driffield".toUpperCase(),"54.005996,-0.443377000000055");
                cities.put("Dronfield".toUpperCase(),"53.30227900000001,-1.4679590000000644");
                cities.put("Droylsden".toUpperCase(),"53.480784,-2.148716000000036");
                cities.put("Dudley".toUpperCase(),"52.512255,-2.081111999999962");
                cities.put("Dufftown".toUpperCase(),"57.443623,-3.128379999999993");
                cities.put("Dukinfield".toUpperCase(),"53.478764,-2.094522999999981");
                cities.put("Dumbarton".toUpperCase(),"55.945287,-4.564554000000044");
                cities.put("Dumfries".toUpperCase(),"55.070859,-3.6051199999999426");
                cities.put("Dunbar".toUpperCase(),"56.002087,-2.5167370000000346");
                cities.put("Dunblane".toUpperCase(),"56.183877,-3.967448999999988");
                cities.put("Dundee".toUpperCase(),"56.462018,-2.970721000000026");
                cities.put("Dunfermline".toUpperCase(),"56.071741,-3.452150999999958");
                cities.put("Dungannon".toUpperCase(),"54.5082725,-6.7669352000000345");
                cities.put("Dunoon".toUpperCase(),"55.950973,-4.926213999999959");
                cities.put("Duns".toUpperCase(),"55.777794,-2.343460999999934");
                cities.put("Dunstable".toUpperCase(),"51.885644,-0.5203900000000203");
                cities.put("Durham".toUpperCase(),"54.77525,-1.5848519999999553");
                cities.put("Dursley".toUpperCase(),"51.678768,-2.3505179999999655");
                cities.put("Easingwold".toUpperCase(),"54.12144079999999,-1.1918739999999843");
                cities.put("East Grinstead".toUpperCase(),"51.128742,-0.014467999999965286");
                cities.put("East Kilbride".toUpperCase(),"55.76435240000001,-
4.1769987999999785");
                cities.put("Eastbourne".toUpperCase(),"50.768035,0.2904720000000225");
                cities.put("Eastleigh".toUpperCase(),"50.967182,-1.374687999999992");
                cities.put("Eastwood".toUpperCase(),"53.01811499999999,-1.3085439999999835");
                cities.put("Ebbw Vale".toUpperCase(),"51.777532,-3.206150999999977");
                cities.put("Edenbridge".toUpperCase(),"51.196259,0.06545099999993909");
                cities.put("Edinburgh".toUpperCase(),"55.953252,-3.188266999999996");
                cities.put("Egham".toUpperCase(),"51.428825,-0.5478759999999738");
                cities.put("Elgin".toUpperCase(),"57.64945400000001,-3.3184850000000097");
                cities.put("Ellesmere Port".toUpperCase(),"53.279812,-2.8974040000000514");
                cities.put("Ellesmere".toUpperCase(),"52.906718,-2.9003380000000334");
                cities.put("Ely".toUpperCase(),"52.399539,0.2623630000000503");
                cities.put("Enniskillen".toUpperCase(),"54.34382429999999,-7.631533600000012");
                cities.put("Epping".toUpperCase(),"51.700328,0.10865499999999884");
                cities.put("Epsom".toUpperCase(),"51.336036,-0.2673819999999978");
                cities.put("Erith".toUpperCase(),"51.480818,0.17467499999997926");
                cities.put("Esher".toUpperCase(),"51.369487,-0.36592700000005607");
                cities.put("Evesham".toUpperCase(),"52.09214900000001,-1.946770000000015");
```

```
                cities.put("Exeter".toUpperCase(),"50.718412,-3.5338990000000194");
                cities.put("Exmouth".toUpperCase(),"50.619957,-3.4137020000000575");
                cities.put("Eye".toUpperCase(),"52.319605,1.146208999999999");
                cities.put("Eyemouth".toUpperCase(),"55.869058,-2.09106799999995");
                cities.put("Failsworth".toUpperCase(),"53.508131,-2.164444000000003");
                cities.put("Fairford".toUpperCase(),"51.70753699999999,-1.7851349999999684");
                cities.put("Fakenham".toUpperCase(),"52.831333,0.8493240000000242");
                cities.put("Falkirk".toUpperCase(),"56.00187750000001,-3.7839131000000634");
                cities.put("Falkland".toUpperCase(),"56.256285,-3.206429000000071");
                cities.put("Falmouth".toUpperCase(),"50.15257099999999,-5.066270000000031");
                cities.put("Fareham".toUpperCase(),"50.8548464,-1.1865867999999864");
                cities.put("Faringdon".toUpperCase(),"51.658477,-1.584679000000051");
                cities.put("Farnborough".toUpperCase(),"51.2868939,-0.7526149999999916");
                cities.put("Farnham".toUpperCase(),"51.214321,-0.7988020000000233");
                cities.put("Farnworth".toUpperCase(),"53.545838,-2.4039599999999837");
                cities.put("Faversham".toUpperCase(),"51.315994,0.8893580000000156");
                cities.put("Felixstowe".toUpperCase(),"51.961726,1.3512550000000374");
                cities.put("Ferndown".toUpperCase(),"50.807364,-1.8997759999999744");
                cities.put("Filey".toUpperCase(),"54.210076,-0.29059400000005553");
                cities.put("Fintona".toUpperCase(),"54.4967772,-7.3146074999999655");
                cities.put("Fishguard".toUpperCase(),"51.993927,-4.975989000000027");
                cities.put("Fivemiletown".toUpperCase(),"54.3775293,-7.317595100000062");
                cities.put("Fleet".toUpperCase(),"51.277283,-0.8426550000000361");
                cities.put("Fleetwood".toUpperCase(),"53.916661,-3.0356729999999743");
                cities.put("Flint".toUpperCase(),"53.1668658,-3.1418908000000556");
                cities.put("Flitwick".toUpperCase(),"52.004605,-0.4979452999999694");
                cities.put("Folkestone".toUpperCase(),"51.081397,1.1694559999999683");
                cities.put("Fordingbridge".toUpperCase(),"50.9258359,-1.7925259999999525");
                cities.put("Forfar".toUpperCase(),"56.643558,-2.889061999999967");
                cities.put("Forres".toUpperCase(),"57.60979099999999,-3.6199799999999414");
                cities.put("Fort William".toUpperCase(),"56.81981700000001,-
5.105218000000036");
                cities.put("Fowey".toUpperCase(),"50.33499,-4.636525000000006");
                cities.put("Framlingham".toUpperCase(),"52.222147,1.3421049999999468");
                cities.put("Fraserburgh".toUpperCase(),"57.693352,-2.0076309999999467");
                cities.put("Frodsham".toUpperCase(),"53.2967045,-2.7245642999999973");
                cities.put("Frome".toUpperCase(),"51.230751,-2.320096000000035");
                cities.put("Gainsborough".toUpperCase(),"53.400575,-0.7744649999999638");
                cities.put("Galashiels".toUpperCase(),"55.623728,-2.814448999999968");
                cities.put("Gateshead".toUpperCase(),"54.95268,-1.603411000000051");
                cities.put("Gillingham".toUpperCase(),"51.386322,0.5514379999999619");
                cities.put("Glasgow".toUpperCase(),"55.864237,-4.251805999999988");
                cities.put("Glastonbury".toUpperCase(),"51.147427,-2.718454000000065");
                cities.put("Glossop".toUpperCase(),"53.4433284,-1.9489069999999629");
                cities.put("Gloucester".toUpperCase(),"51.8642449,-2.2381560000000036");
                cities.put("Godalming".toUpperCase(),"51.18573199999999,-0.6128079999999727");
                cities.put("Godmanchester".toUpperCase(),"52.319427,-0.17516599999999016");
                cities.put("Goole".toUpperCase(),"53.702941,-0.8763810000000376");
                cities.put("Gorseinon".toUpperCase(),"51.669543,-4.041545000000042");
                cities.put("Gosport".toUpperCase(),"50.794995,-1.1175470000000587");
                cities.put("Gourock".toUpperCase(),"55.9591984,-4.8168799000000035");
                cities.put("Grange over Sands".toUpperCase(),"54.191009,-2.9106879999999364");
                cities.put("Grangemouth".toUpperCase(),"56.0097152,-3.7227698000000373");
                cities.put("Grantham".toUpperCase(),"52.912524,-0.6435820000000376");
                cities.put("Grantown on Spey".toUpperCase(),"57.33041,-3.6096279999999297");
                cities.put("Gravesend".toUpperCase(),"51.44188399999999,0.37075900000002093");
                cities.put("Grays".toUpperCase(),"51.4784037,0.32301510000002054");
                cities.put("Great Yarmouth".toUpperCase(),"52.598233,1.7280470000000605");
                cities.put("Greenock".toUpperCase(),"55.95647599999999,-4.771982999999977");
                cities.put("Grimsby".toUpperCase(),"53.567471,-0.0807839999999942");
                cities.put("Guildford".toUpperCase(),"51.23622,-0.5704090000000406");
                cities.put("Haddington".toUpperCase(),"55.95867399999999,-2.7748639999999796");
                cities.put("Hadleigh".toUpperCase(),"52.0450779,0.95264700000007");
                cities.put("Hailsham".toUpperCase(),"50.864612,0.2552339999999731");
                cities.put("Halesowen".toUpperCase(),"52.449845,-2.0505259999999907");
```

```
                cities.put("Halesworth".toUpperCase(),"52.343264,1.5026420000000371");
                cities.put("Halifax".toUpperCase(),"53.72702,-1.8575399999999718");
                cities.put("Halstead".toUpperCase(),"51.945073,0.6390320000000429");
                cities.put("Haltwhistle".toUpperCase(),"54.972237,-2.4608560000000352");
                cities.put("Hamilton".toUpperCase(),"55.77763299999999,-4.053678999999988");
                cities.put("Harlow".toUpperCase(),"51.767787,0.0878060000000005");
                cities.put("Harpenden".toUpperCase(),"51.81845999999999,-0.35895300000004227");
                cities.put("Harrogate".toUpperCase(),"53.99212,-1.5418119999999362");
                cities.put("Hartlepool".toUpperCase(),"54.691745,-1.2129260000000386");
                cities.put("Harwich".toUpperCase(),"51.934731,1.2602970000000369");
                cities.put("Haslemere".toUpperCase(),"51.090856,-0.7133730000000469");
                cities.put("Hastings".toUpperCase(),"50.854259,0.5734529999999722");
                cities.put("Hatfield".toUpperCase(),"51.763366,-0.2230899999999565");
                cities.put("Havant".toUpperCase(),"50.8518324,-0.9847131999999874");
                cities.put("Haverfordwest".toUpperCase(),"51.800475,-4.971318999999994");
                cities.put("Haverhill".toUpperCase(),"52.082766,0.4409450000000561");
                cities.put("Hawarden".toUpperCase(),"53.18569,-3.029807500000061");
                cities.put("Hawick".toUpperCase(),"55.42706,-2.780913999999939");
                cities.put("Hay on Wye".toUpperCase(),"52.075697,-3.1259079999999813");
                cities.put("Hayle".toUpperCase(),"50.185467,-5.420910000000049");
                cities.put("Haywards Heath".toUpperCase(),"50.999041,-0.1063329999999496");
                cities.put("Heanor".toUpperCase(),"53.013807,-1.3537699999999404");
                cities.put("Heathfield".toUpperCase(),"50.9667401,0.2564428000000589");
                cities.put("Hebden Bridge".toUpperCase(),"53.74330399999999,-
2.0130209999999806");
                cities.put("Helensburgh".toUpperCase(),"56.002318,-4.734014000000002");
                cities.put("Helston".toUpperCase(),"50.101593,-5.274995999999987");
                cities.put("Hemel Hempstead".toUpperCase(),"51.753241,-0.44863199999997505");
                cities.put("Henley on Thames".toUpperCase(),"51.535764,-0.9028940000000603");
                cities.put("Hereford".toUpperCase(),"52.05639799999999,-2.71597399999996");
                cities.put("Herne Bay".toUpperCase(),"51.370922,1.1277709999999388");
                cities.put("Hertford".toUpperCase(),"51.795756,-0.08115699999996195");
                cities.put("Hessle".toUpperCase(),"53.723805,-0.4348579999999629");
                cities.put("Heswall".toUpperCase(),"53.3284285,-3.098730400000022");
                cities.put("Hexham".toUpperCase(),"54.972665,-2.112143899999978");
                cities.put("High Wycombe".toUpperCase(),"51.628611,-0.7482290000000376");
                cities.put("Higham Ferrers".toUpperCase(),"52.307374,-0.5928409000000556");
                cities.put("Highworth".toUpperCase(),"51.632861,-1.710397999999941");
                cities.put("Hinckley".toUpperCase(),"52.5454549,-1.37666999999999");
                cities.put("Hitchin".toUpperCase(),"51.94921,-0.2834139999999934");
                cities.put("Hoddesdon".toUpperCase(),"51.76000699999999,-
0.015041999999994005");
                cities.put("Holmfirth".toUpperCase(),"53.571744,-1.786292000000003");
                cities.put("Holsworthy".toUpperCase(),"50.80921499999999,-4.353992999999946");
                cities.put("Holyhead".toUpperCase(),"53.309441,-4.633037999999942");
                cities.put("Holywell".toUpperCase(),"53.27600899999999,-3.22510299999999");
                cities.put("Honiton".toUpperCase(),"50.799468,-3.188682999999969");
                cities.put("Horley".toUpperCase(),"51.173516,-0.17210899999997764");
                cities.put("Horncastle".toUpperCase(),"53.207307,-0.11244999999996708");
                cities.put("Hornsea".toUpperCase(),"53.9104,-0.17395299999998315");
                cities.put("Horsham".toUpperCase(),"51.062883,-0.3258580000000393");
                cities.put("Horwich".toUpperCase(),"53.598398,-2.5547619999999824");
                cities.put("Houghton le Spring".toUpperCase(),"54.841016,-1.4686910000000353");
                cities.put("Hove".toUpperCase(),"50.8279319,-0.1687489999999343");
                cities.put("Howden".toUpperCase(),"53.745894,-0.8688769999999977");
                cities.put("Hoylake".toUpperCase(),"53.393238,-3.1762800999999854");
                cities.put("Hucknall".toUpperCase(),"53.034152,-1.2028900000000249");
                cities.put("Huddersfield".toUpperCase(),"53.645792,-1.7850349999999935");
                cities.put("Hungerford".toUpperCase(),"51.41234499999999,-1.5179950000000417");
                cities.put("Hunstanton".toUpperCase(),"52.9389139,0.49103200000001834");
                cities.put("Huntingdon".toUpperCase(),"52.33146,-0.18255199999998695");
                cities.put("Huntly".toUpperCase(),"57.445936,-2.7878058999999666");
                cities.put("Hyde".toUpperCase(),"53.452977,-2.082785999999942");
                cities.put("Hythe".toUpperCase(),"51.071739,1.081937000000039");
                cities.put("Ilford".toUpperCase(),"51.556619,0.07625099999995655");
```

```
                cities.put("Ilfracombe".toUpperCase(),"51.205163,-4.126761999999985");
                cities.put("Ilkeston".toUpperCase(),"52.970142,-1.3069779999999582");
                cities.put("Ilkley".toUpperCase(),"53.925486,-1.8228169999999864");
                cities.put("Ilminster".toUpperCase(),"50.928494,-2.9110620000000154");
                cities.put("Innerleithen".toUpperCase(),"55.624015,-3.0649120000000494");
                cities.put("Inveraray".toUpperCase(),"56.2309517,-5.077101800000037");
                cities.put("Inverkeithing".toUpperCase(),"56.030043,-3.3987950000000637");
                cities.put("Inverness".toUpperCase(),"57.477773,-4.224721000000045");
                cities.put("Inverurie".toUpperCase(),"57.28347400000001,-2.3739950000000363");
                cities.put("Ipswich".toUpperCase(),"52.056736,1.1482200000000375");
                cities.put("Irthlingborough".toUpperCase(),"52.32559999999999,-
0.6130590000000211");
                cities.put("Irvine".toUpperCase(),"55.61156690000001,-4.669636399999945");
                cities.put("Ivybridge".toUpperCase(),"50.390202,-3.9204310000000078");
                cities.put("Jarrow".toUpperCase(),"54.980297,-1.4827569999999923");
                cities.put("Jedburgh".toUpperCase(),"55.477721,-2.5549369000000297");
                cities.put("Johnstone".toUpperCase(),"55.837324,-4.513914999999997");
                cities.put("Keighley".toUpperCase(),"53.867795,-1.9123580000000402");
                cities.put("Keith".toUpperCase(),"57.543094,-2.951526000000058");
                cities.put("Kelso".toUpperCase(),"55.600029,-2.432207000000062");
                cities.put("Kempston".toUpperCase(),"52.11791479999999,-0.4967914999999721");
                cities.put("Kendal".toUpperCase(),"54.32800599999999,-2.7462900000000445");
                cities.put("Kenilworth".toUpperCase(),"52.349557,-1.5807300000000168");
                cities.put("Kesgrave".toUpperCase(),"52.06256699999999,1.2339070000000447");
                cities.put("Keswick".toUpperCase(),"54.601276,-3.134706000000051");
                cities.put("Kettering".toUpperCase(),"52.396322,-0.730248999999958");
                cities.put("Keynsham".toUpperCase(),"51.415059,-2.502525999999989");
                cities.put("Kidderminster".toUpperCase(),"52.388596,-2.2496839000000364");
                cities.put("Kilbarchan".toUpperCase(),"55.838112,-4.554188999999951");
                cities.put("Kilkeel".toUpperCase(),"54.0631154,-6.007004000000052");
                cities.put("Killyleagh".toUpperCase(),"54.3999861,-5.650795399999993");
                cities.put("Kilmarnock".toUpperCase(),"55.614719,-4.49879199999998");
                cities.put("Kilwinning".toUpperCase(),"55.65591999999999,-4.70311700000002");
                cities.put("Kinghorn".toUpperCase(),"56.071231,-3.1743289999999433");
                cities.put("Kingsbridge".toUpperCase(),"50.283948,-3.7774930000000495");
                cities.put("Kington".toUpperCase(),"52.203742,-3.029047999999989");
                cities.put("Kingussie".toUpperCase(),"57.081409,-4.054468000000043");
                cities.put("Kinross".toUpperCase(),"56.206132,-3.422900000000027");
                cities.put("Kintore".toUpperCase(),"57.233247,-2.346088000000009");
                cities.put("Kirkby".toUpperCase(),"53.48121,-2.891012000000046");
                cities.put("Kirkby Lonsdale".toUpperCase(),"54.204919,-2.6017100000000255");
                cities.put("Kirkcaldy".toUpperCase(),"56.1168249,-3.1581370000000106");
                cities.put("Kirkcudbright".toUpperCase(),"54.83756899999999,-
4.048779999999965");
                cities.put("Kirkham".toUpperCase(),"53.78216399999999,-2.871764999999982");
                cities.put("Kirkwall".toUpperCase(),"58.98467400000001,-2.962248999999929");
                cities.put("Kirriemuir".toUpperCase(),"56.675117,-3.0035010000000284");
                cities.put("Knaresborough".toUpperCase(),"54.011022,-1.4710199999999531");
                cities.put("Knighton".toUpperCase(),"52.341009,-3.046970999999985");
                cities.put("Knutsford".toUpperCase(),"53.3005939,-2.371833000000038");
                cities.put("Ladybank".toUpperCase(),"56.280831,-3.1267100000000028");
                cities.put("Lampeter".toUpperCase(),"52.112903,-4.07850899999994");
                cities.put("Lanark".toUpperCase(),"55.673865,-3.7821380000000318");
                cities.put("Lancaster".toUpperCase(),"54.046575,-2.8007399000000532");
                cities.put("Langholm".toUpperCase(),"55.153062,-2.998913000000016");
                cities.put("Largs".toUpperCase(),"55.79333500000001,-4.867277999999942");
                cities.put("Larne".toUpperCase(),"54.8578003,-5.8236223999999766");
                cities.put("Laugharne".toUpperCase(),"51.769203,-4.464251999999988");
                cities.put("Launceston".toUpperCase(),"50.640134,-4.358558000000016");
                cities.put("Laurencekirk".toUpperCase(),"56.831345,-2.472517000000039");
                cities.put("Leamington Spa".toUpperCase(),"52.2851905,-1.520078900000044");
                cities.put("Leatherhead".toUpperCase(),"51.29640699999999,-
0.33112000000005537");
                cities.put("Ledbury".toUpperCase(),"52.033882,-2.4235740000000305");
                cities.put("Leeds".toUpperCase(),"53.8007554,-1.5490773999999874");
```

```
                cities.put("Leek".toUpperCase(),"53.10915199999999,-2.023392999999942");
                cities.put("Leicester".toUpperCase(),"52.6368778,-1.1397591999999577");
                cities.put("Leighton Buzzard".toUpperCase(),"51.9196839,-0.6606570000000147");
                cities.put("Leiston".toUpperCase(),"52.209044,1.5740799999999808");
                cities.put("Leominster".toUpperCase(),"52.2257,-2.742772000000059");
                cities.put("Lerwick".toUpperCase(),"60.1529871,-1.1492931999999882");
                cities.put("Letchworth".toUpperCase(),"51.979074,-0.22662400000001526");
                cities.put("Leven".toUpperCase(),"56.19632,-2.9965779999999995");
                cities.put("Lewes".toUpperCase(),"50.87387200000001,0.008780000000001564");
                cities.put("Leyland".toUpperCase(),"53.697931,-2.69547799999998");
                cities.put("Lichfield".toUpperCase(),"52.681602,-1.831672000000026");
                cities.put("Limavady".toUpperCase(),"55.0454563,-6.933675799999946");
                cities.put("Lincoln".toUpperCase(),"53.230688,-0.5405789999999797");
                cities.put("Linlithgow".toUpperCase(),"55.9716266,-3.602584699999966");
                cities.put("Lisburn".toUpperCase(),"54.516246,-6.058010599999989");
                cities.put("Liskeard".toUpperCase(),"50.45552,-4.464719000000059");
                cities.put("Lisnaskea".toUpperCase(),"54.2528935,-7.4428695999999945");
                cities.put("Littlehampton".toUpperCase(),"50.811057,-0.538660899999968");
                cities.put("Liverpool".toUpperCase(),"53.4083714,-2.9915726000000404");
                cities.put("Llandeilo".toUpperCase(),"51.88469,-3.9914300000000367");
                cities.put("Llandovery".toUpperCase(),"51.997395,-3.7975079999999934");
                cities.put("Llandrindod Wells".toUpperCase(),"52.24169999999999,-
3.377742000000012");
                cities.put("Llandudno".toUpperCase(),"53.324061,-3.8276089999999385");
                cities.put("Llanelli".toUpperCase(),"51.680886,-4.160248000000024");
                cities.put("Llanfyllin".toUpperCase(),"52.76623499999999,-3.275786000000039");
                cities.put("Llangollen".toUpperCase(),"52.969215,-3.1716599999999744");
                cities.put("Llanidloes".toUpperCase(),"52.44779399999999,-3.5401950000000397");
                cities.put("Llanrwst".toUpperCase(),"53.137033,-3.7957320000000436");
                cities.put("Llantrisant".toUpperCase(),"51.5429188,-3.374952500000063");
                cities.put("Llantwit Major".toUpperCase(),"51.409259,-3.485245999999961");
                cities.put("Llanwrtyd Wells".toUpperCase(),"52.107586,-3.6374270000000024");
                cities.put("Loanhead".toUpperCase(),"55.876406,-3.1493540000000166");
                cities.put("Lochgilphead".toUpperCase(),"56.03829200000001,-
5.432344999999941");
                cities.put("Lockerbie".toUpperCase(),"55.12224500000001,-3.349008000000026");
                cities.put("London".toUpperCase(),"51.5287336,-0.3824693");
                cities.put("Londonderry".toUpperCase(),"54.9966124,-7.308574799999974");
                cities.put("Long Eaton".toUpperCase(),"52.898446,-1.2697779999999739");
                cities.put("Longridge".toUpperCase(),"53.831915,-2.598801999999978");
                cities.put("Looe".toUpperCase(),"50.3562269,-4.455232000000024");
                cities.put("Lossiemouth".toUpperCase(),"57.72157899999999,-3.280324999999948");
                cities.put("Lostwithiel".toUpperCase(),"50.406021,-4.675067899999931");
                cities.put("Loughborough".toUpperCase(),"52.772099,-1.2061659999999392");
                cities.put("Loughton".toUpperCase(),"51.655942,0.06816100000003189");
                cities.put("Louth".toUpperCase(),"53.365962,-0.007710999999972046");
                cities.put("Lowestoft".toUpperCase(),"52.48113799999999,1.753449000000046");
                cities.put("Ludlow".toUpperCase(),"52.367749,-2.7139128999999684");
                cities.put("Lurgan".toUpperCase(),"54.4635261,-6.334550600000057");
                cities.put("Luton".toUpperCase(),"51.8786707,-0.4200255000000652");
                cities.put("Lutterworth".toUpperCase(),"52.45599499999999,-
1.1991590000000087");
                cities.put("Lydd".toUpperCase(),"50.950945,0.9065891999999849");
                cities.put("Lydney".toUpperCase(),"51.72913,-2.5305040000000645");
                cities.put("Lyme Regis".toUpperCase(),"50.725156,-2.9366390000000138");
                cities.put("Lymington".toUpperCase(),"50.758531,-1.5419099000000642");
                cities.put("Lynton".toUpperCase(),"51.2296539,-3.8401020000000017");
                cities.put("Mablethorpe".toUpperCase(),"53.34088200000001,0.2610710000000154");
                cities.put("Macclesfield".toUpperCase(),"53.258663,-2.1192869999999857");
                cities.put("Machynlleth".toUpperCase(),"52.590273,-3.853484999999978");
                cities.put("Maesteg".toUpperCase(),"51.60856200000001,-3.660466000000042");
                cities.put("Magherafelt".toUpperCase(),"54.75537300000001,-6.60798049999994");
                cities.put("Maidenhead".toUpperCase(),"51.522414,-0.7219000000000051");
                cities.put("Maidstone".toUpperCase(),"51.270363,0.5226989999999887");
                cities.put("Maldon".toUpperCase(),"51.73180499999999,0.6714480000000549");
```

```
                cities.put("Malmesbury".toUpperCase(),"51.586357,-2.10283400000003");
                cities.put("Malton".toUpperCase(),"54.136836,-0.7978970000000345");
                cities.put("Malvern".toUpperCase(),"52.1366184,-2.3199773999999707");
                cities.put("Manchester".toUpperCase(),"53.4807593,-2.2426305000000184");
                cities.put("Manningtree".toUpperCase(),"51.945407,1.062086000000022");
                cities.put("Mansfield".toUpperCase(),"53.147195,-1.1986739999999827");
                cities.put("March".toUpperCase(),"52.551716,0.08862199999998666");
                cities.put("Margate".toUpperCase(),"51.38964600000001,1.386833900000056");
                cities.put("Market Deeping".toUpperCase(),"52.6795904,-0.3216218000000026");
                cities.put("Market Drayton".toUpperCase(),"52.903552,-2.4834849999999733");
                cities.put("Market Harborough".toUpperCase(),"52.475769,-0.9215169999999944");
                cities.put("Market Rasen".toUpperCase(),"53.387762,-0.33328499999993255");
                cities.put("Market Weighton".toUpperCase(),"53.8648392,-0.6677465000000211");
                cities.put("Markethill".toUpperCase(),"54.2969635,-6.521738300000038");
                cities.put("Markinch".toUpperCase(),"56.203595,-3.1317688999999973");
                cities.put("Marlborough".toUpperCase(),"51.420073,-1.7270619999999326");
                cities.put("Marlow".toUpperCase(),"51.5719443,-0.7769422000000077");
                cities.put("Maryport".toUpperCase(),"54.714441,-3.4949400000000423");
                cities.put("Matlock".toUpperCase(),"53.137156,-1.551774000000023");
                cities.put("Maybole".toUpperCase(),"55.35302,-4.679055000000062");
                cities.put("Melksham".toUpperCase(),"51.370447,-2.137629000000061");
                cities.put("Melrose".toUpperCase(),"55.598676,-2.72190999999998");
                cities.put("Melton Mowbray".toUpperCase(),"52.766404,-0.8871259999999666");
                cities.put("Merthyr Tydfil".toUpperCase(),"51.74872999999999,-
3.381646000000046");
                cities.put("Mexborough".toUpperCase(),"53.493099,-1.2804519999999684");
                cities.put("Middleham".toUpperCase(),"54.285318,-1.8056301000000303");
                cities.put("Middlesbrough".toUpperCase(),"54.574227,-1.234956000000011");
                cities.put("Middlewich".toUpperCase(),"53.19256499999999,-2.4438330000000406");
                cities.put("Midhurst".toUpperCase(),"50.9868979,-0.7372740000000704");
                cities.put("Midsomer Norton".toUpperCase(),"51.285199,-2.485936000000038");
                cities.put("Milford Haven".toUpperCase(),"51.714306,-5.042696999999976");
                cities.put("Milngavie".toUpperCase(),"55.943304,-4.31698700000004");
                cities.put("Milton Keynes".toUpperCase(),"52.0406224,-0.7594171000000642");
                cities.put("Minehead".toUpperCase(),"51.20428,-3.481152000000655");
                cities.put("Moffat".toUpperCase(),"55.33520799999999,-3.4403369999999995");
                cities.put("Mold".toUpperCase(),"53.167203,-3.1419029999999566");
                cities.put("Monifieth".toUpperCase(),"56.480177,-2.81928199999993");
                cities.put("Monmouth".toUpperCase(),"51.8116533,-2.7163044999999784");
                cities.put("Montgomery".toUpperCase(),"52.56279,-3.149330999999961");
                cities.put("Montrose".toUpperCase(),"56.706922,-2.466114999999945");
                cities.put("Morecambe".toUpperCase(),"54.074166,-2.864967999999976");
                cities.put("Moreton in Marsh".toUpperCase(),"51.9914181,-1.7028526000000284");
                cities.put("Moretonhampstead".toUpperCase(),"50.659678,-3.767892999999958");
                cities.put("Morley".toUpperCase(),"53.744513,-1.5980449999999564");
                cities.put("Morpeth".toUpperCase(),"55.16875,-1.6874930000000177");
                cities.put("Motherwell".toUpperCase(),"55.78320919999999,-3.981096799999932");
                cities.put("Musselburgh".toUpperCase(),"55.941941,-3.053918000000067");
                cities.put("Nailsea".toUpperCase(),"51.43175600000001,-2.7562869999999293");
                cities.put("Nailsworth".toUpperCase(),"51.693878,-2.2162929999999506");
                cities.put("Nairn".toUpperCase(),"57.58642200000001,-3.8684749999999894");
                cities.put("Nantwich".toUpperCase(),"53.06718499999999,-2.5241019999999708");
                cities.put("Narberth".toUpperCase(),"51.799763,-4.744008000000008");
                cities.put("Neath".toUpperCase(),"51.656991,-3.8054759999999987");
                cities.put("Needham Market".toUpperCase(),"52.155608,1.049566000000027");
                cities.put("Neston".toUpperCase(),"53.290584,-3.06910199999993");
                cities.put("New Mills".toUpperCase(),"53.3678916,-2.0047713999999814");
                cities.put("New Milton".toUpperCase(),"50.7531235,-1.6550806000000193");
                cities.put("Newbury".toUpperCase(),"51.401409,-1.323113899999953");
                cities.put("Newcastle Emlyn".toUpperCase(),"52.037442,-4.468710999999985");
                cities.put("Newcastle upon Tyne".toUpperCase(),"54.978252,-
1.6177800000000389");
                cities.put("Newcastle".toUpperCase(),"54.978252,-1.6177800000000389");
                cities.put("Newent".toUpperCase(),"51.929281,-2.404578000000015");
                cities.put("Newhaven".toUpperCase(),"50.79307000000001,0.04557399999998779");
```

```
                cities.put("Newmarket".toUpperCase(),"52.24487999999999,0.4079619999999977");
                cities.put("Newport Pagnell".toUpperCase(),"52.084585,-0.7345829999999296");
                cities.put("Newport".toUpperCase(),"51.584151,-2.997663999999986");
                cities.put("Newport on Tay".toUpperCase(),"56.440473,-2.9407320000000254");
                cities.put("Newquay".toUpperCase(),"50.41549699999999,-5.073718999999983");
                cities.put("Newry".toUpperCase(),"54.1751024,-6.34022989999994");
                cities.put("Newton Abbot".toUpperCase(),"50.52890499999999,-
3.6083599999999948");
                cities.put("Newton Aycliffe".toUpperCase(),"54.61598799999999,-
1.5755770000000666");
                cities.put("Newton Stewart".toUpperCase(),"54.96029900000001,-
4.483163999999988");
                cities.put("Newton le Willows".toUpperCase(),"53.452889,-2.6350800000000163");
                cities.put("Newtown".toUpperCase(),"52.51211199999999,-3.3131060000000616");
                cities.put("Newtownabbey".toUpperCase(),"54.685279,-5.9645024999999805");
                cities.put("Newtownards".toUpperCase(),"54.5913787,-5.69317030000002");
                cities.put("Normanton".toUpperCase(),"53.700876,-1.4171479999999974");
                cities.put("North Berwick".toUpperCase(),"56.058363,-2.7196460000000116");
                cities.put("North Walsham".toUpperCase(),"52.822699,1.3859729000000698");
                cities.put("Northallerton".toUpperCase(),"54.337961,-1.4299590000000535");
                cities.put("Northampton".toUpperCase(),"52.240477,-0.902655999999979");
                cities.put("Northwich".toUpperCase(),"53.25868029999999,-2.5181321000000025");
                cities.put("Norwich".toUpperCase(),"52.6308859,1.2973550000000387");
                cities.put("Nottingham".toUpperCase(),"52.95478319999999,-1.1581085999999914");
                cities.put("Nuneaton".toUpperCase(),"52.520489,-1.4653819999999769");
                cities.put("Oakham".toUpperCase(),"52.6695629,-0.7266250000000127");
                cities.put("Oban".toUpperCase(),"56.415157,-5.471046999999999");
                cities.put("Okehampton".toUpperCase(),"50.738308,-4.004263000000037");
                cities.put("Oldbury".toUpperCase(),"52.504923,-2.01590699999997");
                cities.put("Oldham".toUpperCase(),"53.5409298,-2.11136590000001");
                cities.put("Oldmeldrum".toUpperCase(),"57.33513800000001,-2.321806000000038");
                cities.put("Olney".toUpperCase(),"52.15473799999999,-0.7013779000000113");
                cities.put("Omagh".toUpperCase(),"54.5977149,-7.3099595999999565");
                cities.put("Ormskirk".toUpperCase(),"53.568935,-2.8850569999999607");
                cities.put("Orpington".toUpperCase(),"51.374843,0.09421399999996538");
                cities.put("Ossett".toUpperCase(),"53.681091,-1.578877000000034");
                cities.put("Oswestry".toUpperCase(),"52.857148,-3.056412000000023");
                cities.put("Otley".toUpperCase(),"53.905825,-1.6917859999999791");
                cities.put("Oundle".toUpperCase(),"52.48084069999999,-0.4687516000000187");
                cities.put("Oxford".toUpperCase(),"51.7520209,-1.2577263000000585");
                cities.put("Padstow".toUpperCase(),"50.54206199999999,-4.939017000000035");
                cities.put("Paignton".toUpperCase(),"50.43508,-3.5642350000000533");
                cities.put("Painswick".toUpperCase(),"51.7871669,-2.193555999999944");
                cities.put("Paisley".toUpperCase(),"55.847258,-4.440113999999994");
                cities.put("Peebles".toUpperCase(),"55.653071,-3.1936419999999544");
                cities.put("Pembroke".toUpperCase(),"51.674043,-4.908636999999999");
                cities.put("Penarth".toUpperCase(),"51.438798,-3.1735069999999723");
                cities.put("Penicuik".toUpperCase(),"55.830932,-3.224533000000065");
                cities.put("Penistone".toUpperCase(),"53.526141,-1.625532900000053");
                cities.put("Penmaenmawr".toUpperCase(),"53.266577,-3.9206500000000233");
                cities.put("Penrith".toUpperCase(),"54.664097,-2.752707999999984");
                cities.put("Penryn".toUpperCase(),"50.169173,-5.107087999999976");
                cities.put("Penzance".toUpperCase(),"50.118798,-5.537592000000018");
                cities.put("Pershore".toUpperCase(),"52.109994,-2.0748140000000603");
                cities.put("Perth".toUpperCase(),"56.394994,-3.4308379999999943");
                cities.put("Peterborough".toUpperCase(),"52.56949849999999,-
0.2405298999999559");
                cities.put("Peterhead".toUpperCase(),"57.50812299999999,-1.7840659999999389");
                cities.put("Peterlee".toUpperCase(),"54.762441,-1.3289499999999634");
                cities.put("Petersfield".toUpperCase(),"51.007591,-0.9389899999999898");
                cities.put("Petworth".toUpperCase(),"50.9867009,-0.6107242000000497");
                cities.put("Pickering".toUpperCase(),"54.243925,-0.7775530000000117");
                cities.put("Pitlochry".toUpperCase(),"56.704361,-3.729710999999952");
                cities.put("Pittenweem".toUpperCase(),"56214,-2.732285000000047");
                cities.put("Plymouth".toUpperCase(),"50.3754565,-4.14265649999993");
```

```
cities.put("Pocklington".toUpperCase(),"53.9300565,-0.7794059000000289");
cities.put("Polegate".toUpperCase(),"50.824012,0.24381300000004558");
cities.put("Pontefract".toUpperCase(),"53.691688,-1.3086479999999483");
cities.put("Pontypridd".toUpperCase(),"51.60077399999999,-3.3423139999999876");
cities.put("Poole".toUpperCase(),"50.71505,-1.9872480000000223");
cities.put("Port Talbot".toUpperCase(),"51.5946799,-3.7840969999999743");
cities.put("Portadown".toUpperCase(),"54.4203331,-6.454838699999982");
cities.put("Portaferry".toUpperCase(),"54.38173519999999,-5.54698170000006");
cities.put("Porth".toUpperCase(),"51.613554,-3.4071840000000293");
cities.put("Porthcawl".toUpperCase(),"51.47896799999999,-3.7051629999999705");
cities.put("Porthmadog".toUpperCase(),"52.927842,-4.133383999999978");
cities.put("Portishead".toUpperCase(),"51.485155,-2.767947999999933");
cities.put("Portrush".toUpperCase(),"55.2041945,-6.6526708999999755");
cities.put("Portsmouth".toUpperCase(),"50.8197675,-1.0879769000000579");
cities.put("Portstewart".toUpperCase(),"55.1868773,-6.717104100000029");
cities.put("Potters Bar".toUpperCase(),"51.696636,-0.17594800000006217");
cities.put("Potton".toUpperCase(),"52.12706600000001,-0.21519499999999425");
cities.put("Poulton le Fylde".toUpperCase(),"53.843964,-2.9862809999999627");
cities.put("Prescot".toUpperCase(),"53.428674,-2.8045640000000276");
cities.put("Prestatyn".toUpperCase(),"53.336512,-3.4076129999999694");
cities.put("Presteigne".toUpperCase(),"52.27203100000001,-3.005157000000054");
cities.put("Preston".toUpperCase(),"53.763201,-2.7030899999999747");
cities.put("Prestwick".toUpperCase(),"55.495587,-4.6142138999999815");
cities.put("Princes Risborough".toUpperCase(),"51.724376,-0.8345180000000028");
cities.put("Prudhoe".toUpperCase(),"54.962199,-1.8472299999999677");
cities.put("Pudsey".toUpperCase(),"53.795766,-1.6761082000000442");
cities.put("Pwllheli".toUpperCase(),"52.888816,-4.417633900000055");
cities.put("Ramsgate".toUpperCase(),"51.335545,1.4198949999999968");
cities.put("Randalstown".toUpperCase(),"54.75005179999999,-6.32027400000004");
cities.put("Rayleigh".toUpperCase(),"51.586385,0.6048710000000028");
cities.put("Reading".toUpperCase(),"51.4542645,-0.9781302999999753");
cities.put("Redcar".toUpperCase(),"54.5974636,-1.0779515000000401");
cities.put("Redditch".toUpperCase(),"52.30897,-1.9409359999999651");
cities.put("Redhill".toUpperCase(),"51.239208,-0.16988000000003467");
cities.put("Redruth".toUpperCase(),"50.233022,-5.226666000000023");
cities.put("Reigate".toUpperCase(),"51.237276,-0.20588299999997162");
cities.put("Retford".toUpperCase(),"53.32135599999999,-0.9455030000000306");
cities.put("Rhayader".toUpperCase(),"52.301537,-3.510640999999964");
cities.put("Rhuddlan".toUpperCase(),"53.291911,-3.468319000000065");
cities.put("Rhyl".toUpperCase(),"53.31914099999999,-3.4916339999999764");
cities.put("Richmond".toUpperCase(),"51.46131099999999,-0.3037420000000566");
cities.put("Rickmansworth".toUpperCase(),"51.638763,-0.474130999999943");
cities.put("Ringwood".toUpperCase(),"50.844949,-1.7888169999999946");
cities.put("Ripley".toUpperCase(),"53.05281900000001,-1.4057920000000195");
cities.put("Ripon".toUpperCase(),"54.1361346,-1.5237756000000218");
cities.put("Rochdale".toUpperCase(),"53.6097136,-2.1561000000000377");
cities.put("Rochester".toUpperCase(),"51.38800000000001,0.5067209999999704");
cities.put("Rochford".toUpperCase(),"51.582071,0.7065149999999676");
cities.put("Romford".toUpperCase(),"51.577076,0.17831899999998768");
cities.put("Romsey".toUpperCase(),"50.98893,-1.4965799999999945");
cities.put("Ross on Wye".toUpperCase(),"51.91445,-2.5824460000000045");
cities.put("Rostrevor".toUpperCase(),"54.10073,-6.200399999999945");
cities.put("Rothbury".toUpperCase(),"55.310464,-1.909546999999975");
cities.put("Rotherham".toUpperCase(),"53.4326035,-1.3635008999999627");
cities.put("Rothesay".toUpperCase(),"55.835963,-5.055792999999994");
cities.put("Rowley Regis".toUpperCase(),"52.488674,-2.046084999999948");
cities.put("Royston".toUpperCase(),"52.048142,-0.024065999999947962");
cities.put("Rugby".toUpperCase(),"52.370878,-1.2650320000000193");
cities.put("Rugeley".toUpperCase(),"52.761515,-1.935967000000005");
cities.put("Runcorn".toUpperCase(),"53.342078,-2.729673000000048");
cities.put("Rushden".toUpperCase(),"52.289125,-0.6003630000000157");
cities.put("Rutherglen".toUpperCase(),"55.828972,-4.22426889999997");
cities.put("Ruthin".toUpperCase(),"53.114725,-3.3108740000000125");
cities.put("Ryde".toUpperCase(),"50.729952,-1.163281999999981");
cities.put("Rye".toUpperCase(),"50.949708,0.737259999999992");
```

```
cities.put("Saffron Walden".toUpperCase(),"52.022593,0.23921500000005835");
cities.put("Saintfield".toUpperCase(),"54.4601231,-5.836099900000022");
cities.put("Salcombe".toUpperCase(),"50.23758,-3.769790999999941");
cities.put("Sale".toUpperCase(),"53.42556099999999,-2.3237020000000257");
cities.put("Salford".toUpperCase(),"53.48752349999999,-2.290126399999963");
cities.put("Salisbury".toUpperCase(),"51.068785,-1.7944720000000416");
cities.put("Saltash".toUpperCase(),"50.40920999999999,-4.216429999999946");
cities.put("Saltcoats".toUpperCase(),"55.637652,-4.78348989999995");
cities.put("Sandbach".toUpperCase(),"53.146561,-2.3673820000000205");
cities.put("Sandhurst".toUpperCase(),"51.3462419,-0.8042679999999791");
cities.put("Sandown".toUpperCase(),"50.659079,-1.1493970000000218");
cities.put("Sandwich".toUpperCase(),"51.274017,1.3374269999999342");
cities.put("Sandy".toUpperCase(),"52.128118,-0.2867880000000014");
cities.put("Sawbridgeworth".toUpperCase(),"51.81554,0.14720999999997275");
cities.put("Saxmundham".toUpperCase(),"52.215855,1.4880560000000287");
cities.put("Scarborough".toUpperCase(),"54.283113,-0.39975200000003497");
cities.put("Scunthorpe".toUpperCase(),"53.588646,-0.6544129999999768");
cities.put("Seaford".toUpperCase(),"50.7734669,0.10110800000006748");
cities.put("Seaton".toUpperCase(),"50.70531339999999,-3.0718733999999586");
cities.put("Sedgefield".toUpperCase(),"54.6594559,-1.4516576999999415");
cities.put("Selby".toUpperCase(),"53.78352400000001,-1.0671889999999848");
cities.put("Selkirk".toUpperCase(),"55.550658,-2.8385240000000067");
cities.put("Selsey".toUpperCase(),"50.730991,-0.7937039999999342");
cities.put("Settle".toUpperCase(),"54.06824599999999,-2.2776549999999816");
cities.put("Sevenoaks".toUpperCase(),"51.27241000000001,0.19089800000006107");
cities.put("Shaftesbury".toUpperCase(),"51.0046,-2.198082999999997");
cities.put("Shanklin".toUpperCase(),"50.63467499999999,-1.1751990000000205");
cities.put("Sheerness".toUpperCase(),"51.44011,0.7641579999999522");
cities.put("Sheffield".toUpperCase(),"53.38112899999999,-1.47008500000004");
cities.put("Shepshed".toUpperCase(),"52.7701009,-1.2924269999999751");
cities.put("Shepton Mallet".toUpperCase(),"51.1909,-2.5478860000000623");
cities.put("Sherborne".toUpperCase(),"50.947822,-2.5144828999999618");
cities.put("Sheringham".toUpperCase(),"52.94442100000001,1.210958900000037");
cities.put("Shildon".toUpperCase(),"54.63341699999999,-1.65491499999996");
cities.put("Shipston on Stour".toUpperCase(),"52.0606549,-1.6228140000000622");
cities.put("Shoreham by Sea".toUpperCase(),"50.8342086,-0.27155579999998736");
cities.put("Shrewsbury".toUpperCase(),"52.70730289999999,-2.7553268000000344");
cities.put("Sidmouth".toUpperCase(),"50.67865,-3.2375600000000304");
cities.put("Sittingbourne".toUpperCase(),"51.340402,0.7315959999999677");
cities.put("Skegness".toUpperCase(),"53.146403,0.3378810000000385");
cities.put("Skelmersdale".toUpperCase(),"53.5503519,-2.7763700000000426");
cities.put("Skipton".toUpperCase(),"53.9628495,-2.01627869999993");
cities.put("Sleaford".toUpperCase(),"53.0003079,-0.409650000000056");
cities.put("Slough".toUpperCase(),"51.51053839999999,-0.5950405999999475");
cities.put("Smethwick".toUpperCase(),"52.492401,-1.965206999999964");
cities.put("Soham".toUpperCase(),"52.335212,0.3375078999999914");
cities.put("Solihull".toUpperCase(),"52.411811,-1.7776099999999815");
cities.put("Somerton".toUpperCase(),"51.0551109,-2.733788000000004");
cities.put("South Molton".toUpperCase(),"51.016818,-3.8321190000000342");
cities.put("South Shields".toUpperCase(),"54.999424,-1.427406000000019");
cities.put("South Woodham Ferrers".toUpperCase(),"51.6465,0.6146650000000591");
cities.put("Southam".toUpperCase(),"52.2505049,-1.3897190000000137");
cities.put("Southampton".toUpperCase(),"50.90970040000001,-1.404350900000054");
cities.put("Southborough".toUpperCase(),"51.1600204,0.25763259999996535");
cities.put("Southend on Sea".toUpperCase(),"51.5459269,0.7077123000000256");
cities.put("Southport".toUpperCase(),"53.645708,-3.0101130000000467");
cities.put("Southsea".toUpperCase(),"50.783565,-1.0855589999999893");
cities.put("Southwell".toUpperCase(),"53.079044,-0.9599210000000085");
cities.put("Southwold".toUpperCase(),"52.32562799999999,1.6801809999999477");
cities.put("Spalding".toUpperCase(),"52.79010160000001,-0.15370239999992918");
cities.put("Spennymoor".toUpperCase(),"54.69772099999999,-1.585519999999974");
cities.put("Spilsby".toUpperCase(),"53.174675,0.09143990000006852");
cities.put("Stafford".toUpperCase(),"52.806693,-2.1206600000000435");
cities.put("Staines".toUpperCase(),"51.43148,-0.5155250000000251");
cities.put("Stamford".toUpperCase(),"52.65128199999999,-0.4802160000000413");
```

```
                cities.put("Stanley".toUpperCase(),"54.868948,-1.6988410000000158");
                cities.put("Staveley".toUpperCase(),"54.377897,-2.8176770000000033");
                cities.put("Stevenage".toUpperCase(),"51.903761,-0.19661199999995915");
                cities.put("Stirling".toUpperCase(),"56.1165227,-3.93690289999995");
                cities.put("Stockport".toUpperCase(),"53.41063159999999,-2.157533199999989");
                cities.put("Stockton on Tees".toUpperCase(),"54.5704551,-1.328982099999962");
                cities.put("Stoke on Trent".toUpperCase(),"53.002668,-2.179403999999977");
                cities.put("Stone".toUpperCase(),"52.907932,-2.1440450000000055");
                cities.put("Stowmarket".toUpperCase(),"52.188902,0.9977119999999786");
                cities.put("Strabane".toUpperCase(),"54.8273865,-7.463310299999989");
                cities.put("Stranraer".toUpperCase(),"54.903367,-5.024054999999976");
                cities.put("Stratford upon Avon".toUpperCase(),"52.19173,-1.7082980000000134");
                cities.put("Strood".toUpperCase(),"51.3932587,0.47534399999995003");
                cities.put("Stroud".toUpperCase(),"51.745734,-2.2177580000000034");
                cities.put("Sudbury".toUpperCase(),"52.041047,0.7267060000000356");
                cities.put("Sunderland".toUpperCase(),"54.906869,-1.3838009999999485");
                cities.put("Sutton Coldfield".toUpperCase(),"52.57038499999999,-
1.824041999999963");
                cities.put("Sutton in Ashfield".toUpperCase(),"53.126933,-1.2625040000000354");
                cities.put("Swadlincote".toUpperCase(),"52.771318,-1.5549969999999576");
                cities.put("Swanage".toUpperCase(),"50.608277,-1.9607690000000275");
                cities.put("Swanley".toUpperCase(),"51.396531,0.17732599999999366");
                cities.put("Swansea".toUpperCase(),"51.62144,-3.9436459999999443");
                cities.put("Swindon".toUpperCase(),"51.55577390000001,-1.7797176000000263");
                cities.put("Tadcaster".toUpperCase(),"53.883551,-1.2608890000000201");
                cities.put("Tadley".toUpperCase(),"51.343801,-1.1321689999999762");
                cities.put("Tain".toUpperCase(),"57.811501,-4.055229999999938");
                cities.put("Talgarth".toUpperCase(),"51.995873,-3.2323260000000573");
                cities.put("Tamworth".toUpperCase(),"52.633584,-1.6910319999999501");
                cities.put("Taunton".toUpperCase(),"51.015344,-3.106849000000011");
                cities.put("Tavistock".toUpperCase(),"50.5511229,-4.141654000000017");
                cities.put("Teignmouth".toUpperCase(),"50.54703300000001,-3.496687999999949");
                cities.put("Telford".toUpperCase(),"52.678419,-2.445257999999967");
                cities.put("Tenby".toUpperCase(),"51.672738,-4.703578999999991");
                cities.put("Tenterden".toUpperCase(),"51.069358,0.6891219999999976");
                cities.put("Tetbury".toUpperCase(),"51.639295,-2.1581800000000158");
                cities.put("Tewkesbury".toUpperCase(),"51.992265,-2.1579600000000028");
                cities.put("Thame".toUpperCase(),"51.746997,-0.9741880000000265");
                cities.put("Thatcham".toUpperCase(),"51.40580500000001,-1.2664680000000317");
                cities.put("Thaxted".toUpperCase(),"51.9532618,0.34461829999997917");
                cities.put("Thetford".toUpperCase(),"52.412856,0.7516570000000229");
                cities.put("Thirsk".toUpperCase(),"54.233849,-1.3413769999999658");
                cities.put("Thornbury".toUpperCase(),"51.608306,-2.5251530000000457");
                cities.put("Thrapston".toUpperCase(),"52.394588,-0.5358589999999595");
                cities.put("Thurso".toUpperCase(),"58.593566,-3.52207999999996");
                cities.put("Tilbury".toUpperCase(),"51.463024,0.36049800000000687");
                cities.put("Tillicoultry".toUpperCase(),"56.153916,-3.740211000000045");
                cities.put("Tipton".toUpperCase(),"52.52625,-2.066057999999998");
                cities.put("Tiverton".toUpperCase(),"50.902049,-3.491207000000313");
                cities.put("Tobermory".toUpperCase(),"56.6227813,-6.072300400000017");
                cities.put("Todmorden".toUpperCase(),"53.716344,-2.098796900000025");
                cities.put("Tonbridge".toUpperCase(),"51.195043,0.27567999999996573");
                cities.put("Torpoint".toUpperCase(),"50.37529,-4.194344000000001");
                cities.put("Torquay".toUpperCase(),"50.4619209,-3.5253149999999778");
                cities.put("Totnes".toUpperCase(),"50.433741,-3.6857969999999796");
                cities.put("Totton".toUpperCase(),"50.9141239,-1.4985934000000043");
                cities.put("Towcester".toUpperCase(),"52.135107,-0.989688000000001");
                cities.put("Tredegar".toUpperCase(),"51.772619,-3.2467750000000706");
                cities.put("Tregaron".toUpperCase(),"52.2199979,-3.9343850000000202");
                cities.put("Tring".toUpperCase(),"51.79607799999999,-0.6558790000000272");
                cities.put("Troon".toUpperCase(),"55.541332,-4.659947999999986");
                cities.put("Trowbridge".toUpperCase(),"51.319664,-2.2088529999999764");
                cities.put("Truro".toUpperCase(),"50.263195,-5.051040999999941");
                cities.put("Tunbridge Wells".toUpperCase(),"51.132377,0.26369499999998425");
                cities.put("Tywyn".toUpperCase(),"52.587006,-4.087248999999929");
```

```
                cities.put("Uckfield".toUpperCase(),"50.966414,0.09591299999999592");
                cities.put("Ulverston".toUpperCase(),"54.195138,-3.0926699999999983");
                cities.put("Uppingham".toUpperCase(),"52.59122499999999,-0.7184839999999895");
                cities.put("Usk".toUpperCase(),"51.703533,-2.903404000000023");
                cities.put("Uttoxeter".toUpperCase(),"52.89811599999999,-1.8658010000000331");
                cities.put("Ventnor".toUpperCase(),"50.595484,-1.2060999999999922");
                cities.put("Verwood".toUpperCase(),"50.8789783,-1.8654487000000017");
                cities.put("Wadebridge".toUpperCase(),"50.516654,-4.836455999999998");
                cities.put("Wadhurst".toUpperCase(),"51.061756,0.33802000000002863");
                cities.put("Wakefield".toUpperCase(),"53.683298,-1.50592400000005");
                cities.put("Wallasey".toUpperCase(),"53.426521,-3.0662150000000565");
                cities.put("Wallingford".toUpperCase(),"51.5974177,-1.1335612999999967");
                cities.put("Walsall".toUpperCase(),"52.586214,-1.982919000000038");
                cities.put("Waltham Abbey".toUpperCase(),"51.685034,0.002723000000059983");
                cities.put("Waltham Cross".toUpperCase(),"51.68584389999999,-
0.03309639999997671");
                cities.put("Walton on Thames".toUpperCase(),"51.38847,-0.4169689999999946");
                cities.put("Walton on the Naze".toUpperCase(),"51.848186,1.2677360000000135");
                cities.put("Wantage".toUpperCase(),"51.588868,-1.4264530000000377");
                cities.put("Ware".toUpperCase(),"51.810437,-0.028177000000027874");
                cities.put("Wareham".toUpperCase(),"50.687817,-2.110981000000038");
                cities.put("Warminster".toUpperCase(),"51.204629,-2.1810780000000705");
                cities.put("Warrenpoint".toUpperCase(),"54.105301,-6.252077800000052");
                cities.put("Warrington".toUpperCase(),"53.3900441,-2.596950099999958");
                cities.put("Warwick".toUpperCase(),"52.28231599999999,-1.5849269999999933");
                cities.put("Washington".toUpperCase(),"54.89743199999999,-1.5173660000000382");
                cities.put("Watford".toUpperCase(),"51.656489,-0.39031999999997424");
                cities.put("Wednesbury".toUpperCase(),"52.552888,-2.02207999999996");
                cities.put("Wednesfield".toUpperCase(),"52.59625699999999,-2.083409999999958");
                cities.put("Wellingborough".toUpperCase(),"52.302419,-0.6939640000000509");
                cities.put("Wellington".toUpperCase(),"50.978564,-3.2244988999999578");
                cities.put("Wells next the Sea".toUpperCase(),"52.954641,0.8489270000000033");
                cities.put("Wells".toUpperCase(),"51.209347,-2.644597900000008");
                cities.put("Welshpool".toUpperCase(),"52.660348,-3.1464069999999538");
                cities.put("Welwyn Garden City".toUpperCase(),"51.8031689,-
0.20866100000000642");
                cities.put("Wem".toUpperCase(),"52.853637,-2.7267120000000205");
                cities.put("Wendover".toUpperCase(),"51.761877,-0.7397789999999986");
                cities.put("West Bromwich".toUpperCase(),"52.517664,-1.9951590000000579");
                cities.put("Westbury".toUpperCase(),"51.256659,-2.1857720000000427");
                cities.put("Westerham".toUpperCase(),"51.266969,0.07182699999998476");
                cities.put("Westhoughton".toUpperCase(),"53.54896,-2.5258810000000267");
                cities.put("Weston super Mare".toUpperCase(),"51.34740499999999,-
2.9772550000000138");
                cities.put("Wetherby".toUpperCase(),"53.92705600000001,-1.3848160000000007");
                cities.put("Weybridge".toUpperCase(),"51.3716269,-0.457903999999985");
                cities.put("Weymouth".toUpperCase(),"50.6144279,-2.4576210000000174");
                cities.put("Whaley Bridge".toUpperCase(),"53.3298746,-1.9841742999999497");
                cities.put("Whitby".toUpperCase(),"54.486335,-0.6133469999999761");
                cities.put("Whitchurch".toUpperCase(),"52.968716,-2.6820450000000164");
                cities.put("Whitehaven".toUpperCase(),"54.549699,-3.5892330000000356");
                cities.put("Whitley Bay".toUpperCase(),"55.046389,-1.4512988999999834");
                cities.put("Whitnash".toUpperCase(),"52.2638136,-1.5201032999999597");
                cities.put("Whitstable".toUpperCase(),"51.361047,1.0242560000000367");
                cities.put("Whitworth".toUpperCase(),"53.660416,-2.1724779999999555");
                cities.put("Wick".toUpperCase(),"58.438936,-3.093715999999972");
                cities.put("Wickford".toUpperCase(),"51.611309,0.5206799999999703");
                cities.put("Widnes".toUpperCase(),"53.361024,-2.7336370000000443");
                cities.put("Wigan".toUpperCase(),"53.5450645,-2.6325074000000086");
                cities.put("Wigston".toUpperCase(),"52.5863179,-1.106353899999931");
                cities.put("Wigtown".toUpperCase(),"54.867159,-4.444234000000051");
                cities.put("Willenhall".toUpperCase(),"52.585017,-2.0576350000000048");
                cities.put("Wincanton".toUpperCase(),"51.055688,-2.416007000000036");
                cities.put("Winchester".toUpperCase(),"51.059771,-1.3101420000000417");
                cities.put("Windermere".toUpperCase(),"54.380685,-2.9067850000000135");
```

```
                cities.put("Winsford".toUpperCase(),"53.19426199999999,-2.519670000000019");
                cities.put("Winslow".toUpperCase(),"51.944859,-0.878737000000001");
                cities.put("Wisbech".toUpperCase(),"52.666317,0.1587970000000496");
                cities.put("Witham".toUpperCase(),"51.7978049,0.637217899999996");

cities.put("Withernsea".toUpperCase(),"53.73125599999999,0.032889000000068336");
                cities.put("Witney".toUpperCase(),"51.7859365,-1.4850543999999672");
                cities.put("Woburn".toUpperCase(),"51.987703,-0.6209208999999873");
                cities.put("Woking".toUpperCase(),"51.316774,-0.5600349000000051");
                cities.put("Wokingham".toUpperCase(),"51.410457,-0.8338610000000699");
                cities.put("Wolverhampton".toUpperCase(),"52.586973,-2.128820000000019");
                cities.put("Wombwell".toUpperCase(),"53.521016,-1.3993189999999913");
                cities.put("Woodbridge".toUpperCase(),"52.095481,1.3125709999999344");
                cities.put("Woodstock".toUpperCase(),"51.847267,-1.3540910000000395");
                cities.put("Wootton Bassett".toUpperCase(),"51.54139499999999,-
1.9023180000000366");
                cities.put("Worcester".toUpperCase(),"52.193636,-2.22157500000003");
                cities.put("Workington".toUpperCase(),"54.643569,-3.542752000000064");
                cities.put("Worksop".toUpperCase(),"53.309302,-1.122745000000009");
                cities.put("Worthing".toUpperCase(),"50.81787,-0.37288200000000415");
                cities.put("Wotton under Edge".toUpperCase(),"51.63802500000001,-
2.3512479999999414");
                cities.put("Wrexham".toUpperCase(),"53.04304,-2.992493999999965");
                cities.put("Wymondham".toUpperCase(),"52.569354,1.1153050000000349");
                cities.put("Yarm".toUpperCase(),"54.504523,-1.3547590000000582");
                cities.put("Yarmouth".toUpperCase(),"52.598233,1.7280470000000605");
                cities.put("Yate".toUpperCase(),"51.54158899999999,-2.4143229999999676");
                cities.put("Yateley".toUpperCase(),"51.3426285,-0.826272900000049");
                cities.put("Yeadon".toUpperCase(),"53.865153,-1.684203000000025");
                cities.put("Yeovil".toUpperCase(),"50.942061,-2.6333079999999427");
                cities.put("York".toUpperCase(),"53.95996510000001,-1.0872979000000669");

        }
}
```

1.7 Libraries

1.7.1 Library “ReverseGeocode”

1.7.1.1 “GeoName” (GeoName.java)

```
/*
The MIT License (MIT)
[OSI Approved License]
The MIT License (MIT)

Copyright (c) 2014 Daniel Glasson

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
```

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
*/

```
package geocode;

import geocode.kdtree.KDNodeComparator;
import static java.lang.Math.cos;
import static java.lang.Math.sin;
import static java.lang.Math.toRadians;

import java.util.Comparator;

/**
 * Created by Daniel Glasson on 18/05/2014.
 * This class works with a place names files from http://download.geonames.org/export/dump/
 */

public class GeoName extends KDNodeComparator<GeoName> {

    public String name;
    public String geoid;

    public boolean majorPlace; // Major or minor place

    public double latitude;
    public double longitude;

    public double point[] = new double[3]; // The 3D coordinates of the point
    public String country;

    GeoName(String data) {

      String[] names = data.split(",");

      name = names[2];
      geoid = "c"+names[0];

   // majorPlace = names[6].equals("P");
      majorPlace = true;

      latitude = Double.parseDouble(names[4]);
      longitude = Double.parseDouble(names[5]);

      setPoint();
      country = names[4];

    }

    GeoName(Double latitude, Double longitude) {
        name = country = "Search";
        this.latitude = latitude;
        this.longitude = longitude;
        setPoint();
    }

    private void setPoint() {
```

```
        point[0] = cos(toRadians(latitude)) * cos(toRadians(longitude));
        point[1] = cos(toRadians(latitude)) * sin(toRadians(longitude));
        point[2] = sin(toRadians(latitude));
    }

    @Override
    public String toString() {
        return name;
    }

    @Override
    protected Double squaredDistance(Object other) {
        GeoName location = (GeoName)other;
        double x = this.point[0] - location.point[0];
        double y = this.point[1] - location.point[1];
        double z = this.point[2] - location.point[2];
        return (x*x) + (y*y) + (z*z);
    }

    @Override
    protected Double axisSquaredDistance(Object other, Integer axis) {
        GeoName location = (GeoName)other;
        Double distance = point[axis] - location.point[axis];
        return distance * distance;
    }

    @Override
    protected Comparator<GeoName> getComparator(Integer axis) {
        return GeoNameComparator.values()[axis];
    }

    protected static enum GeoNameComparator implements Comparator<GeoName> {
        x {
            @Override
            public int compare(GeoName a, GeoName b) {
                return Double.compare(a.point[0], b.point[0]);
            }
        },
        y {
            @Override
            public int compare(GeoName a, GeoName b) {
                return Double.compare(a.point[1], b.point[1]);
            }
        },
        z {
            @Override
            public int compare(GeoName a, GeoName b) {
                return Double.compare(a.point[2], b.point[2]);
            }
        };
    }
}
```

1.7.1.2 “Reverse Geocode” (ReverseGeocode.java)

```
/*
The MIT License (MIT)
[OSI Approved License]
The MIT License (MIT)

Copyright (c) 2014 Daniel Glasson
```

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
*/

```java
package geocode;

import geocode.kdtree.KDTree;
import java.io.*;
import java.util.ArrayList;

/**
 *
 * Created by Daniel Glasson on 18/05/2014.
 * Uses KD-trees to quickly find the nearest point
 *
 * ReverseGeoCode reverseGeoCode = new ReverseGeoCode(new FileInputStream("c:\\AU.txt"), true);
 * System.out.println("Nearest to -23.456, 123.456 is " + geocode.nearestPlace(-23.456,
123.456));
 */
public class ReverseGeoCode {
    KDTree<GeoName> kdTree;

    // Get placenames from http://download.geonames.org/export/dump/
    public ReverseGeoCode( InputStream placenames, Boolean majorOnly ) throws IOException {
        ArrayList<GeoName> arPlaceNames;
        arPlaceNames = new ArrayList<GeoName>();
        // Read the geonames file in the directory
        BufferedReader in = new BufferedReader(new InputStreamReader(placenames));
        String str;
        try {
                in.readLine();
            while ((str = in.readLine()) != null) {
                GeoName newPlace = new GeoName(str);
                if ( !majorOnly || newPlace.majorPlace ) {
                    arPlaceNames.add(new GeoName(str));
                }
            }
        } catch (IOException ex) {
            in.close();
            throw ex;
        }
        in.close();
        kdTree = new KDTree<GeoName>(arPlaceNames);
    }

    public GeoName nearestPlace(double latitude, double longitude) {
        return kdTree.findNearest(new GeoName(latitude,longitude));
    }
}
```

1.7.2 Library “KDTree”

1.7.2.1 KD Tree (KDTree.java)

```
/*
The MIT License (MIT)
[OSI Approved License]
The MIT License (MIT)

Copyright (c) 2014 Daniel Glasson

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
*/

package geocode.kdtree;

import java.util.Collections;
import java.util.List;

/**
 *
 * @author Daniel Glasson
 * A KD-Tree implementation to quickly find nearest points
 * Currently implements createKDTree and findNearest as that's all that's required here
 */
public class KDTree<T extends KDNodeComparator<T>> {
    private KDNode<T> root;

    public KDTree( List<T> items ) {
        root = createKDTree(items, 0);
    }

    public T findNearest( T search ) {
        return findNearest(root, search, 0).location;
    }

    // Only ever goes to log2(items.length) depth so lack of tail recursion is a non-issue
    private KDNode<T> createKDTree( List<T> items, int depth ) {
        if ( items.isEmpty() ) {
            return null;
        }
```

```
        Collections.sort(items, items.get(0).getComparator(depth % 3));
        int currentIndex = items.size()/2;
        return new KDNode<T>(createKDTree(items.subList(0, currentIndex), depth+1),
createKDTree(items.subList(currentIndex + 1, items.size()), depth+1), items.get(currentIndex));
    }

    private KDNode<T> findNearest(KDNode<T> currentNode, T search, int depth) {
        int direction = search.getComparator(depth % 3).compare( search,
currentNode.location );
        KDNode<T> next = (direction < 0) ? currentNode.left : currentNode.right;
        KDNode<T> other = (direction < 0) ? currentNode.right : currentNode.left;
        KDNode<T> best = (next == null) ? currentNode : findNearest(next, search, depth + 1);
// Go to a leaf
        if ( currentNode.location.squaredDistance(search) <
best.location.squaredDistance(search) ) {
            best = currentNode; // Set best as required
        }
        if ( other != null ) {
            if ( currentNode.location.axisSquaredDistance(search, depth % 3) <
best.location.squaredDistance(search) ) {
                KDNode<T> possibleBest = findNearest( other, search, depth + 1 );
                if (  possibleBest.location.squaredDistance(search) <
best.location.squaredDistance(search) ) {
                    best = possibleBest;
                }
            }
        }
        return best; // Work back up
    }
}
```

1.7.2.2 KD Node (KDNode.java)

```
/*
The MIT License (MIT)
[OSI Approved License]
The MIT License (MIT)

Copyright (c) 2014 Daniel Glasson

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
*/

package geocode.kdtree;
```

```java
/**
 *
 * @author Daniel Glasson
 */
public class KDNode<T extends KDNodeComparator<T>> {
    KDNode<T> left;
    KDNode<T> right;
    T location;

    public KDNode( KDNode<T> left, KDNode<T> right, T location ) {
        this.left = left;
        this.right = right;
        this.location = location;
    }
}
```

1.7.2.3 KD Node Comparator (KDNodeComparator.java)

```java
/*
The MIT License (MIT)
[OSI Approved License]
The MIT License (MIT)

Copyright (c) 2014 Daniel Glasson

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
*/

package geocode.kdtree;

import java.util.Comparator;

/**
 *
 * @author Daniel Glasson
 * Make the user return a comparator for each axis
 * Squared distances should be an optimization
 */
public abstract class KDNodeComparator<T> {
    // This should return a comparator for whatever axis is passed in
    protected abstract Comparator<T> getComparator(Integer axis);
```

```
    // Return squared distance between current and other
    protected abstract <T> Double squaredDistance(T other);

    // Return squared distance between one axis only
    protected abstract <T> Double axisSquaredDistance(T other, Integer axis);
}
```

PART 2: VISUALISATION

2.1 Python script to receive messages from REDIS (app.py)

```
from flask import Flask, render_template, Response

import redis

app = Flask(__name__)
r = redis.StrictRedis(host='127.0.0.1', port=6379, db=0)

def event_stream():
    pubsub = r.pubsub()
    pubsub.subscribe('FluDetectorTopology')
    for message in pubsub.listen():
        print message
        yield 'data: %s\n\n' % message['data']

@app.route('/map')
def show_homepage():
    return render_template("map.html")

@app.route('/stream')
def stream():
    return Response(event_stream(), mimetype="text/event-stream")


if __name__ == '__main__':
    app.run(threaded=True,
    host='0.0.0.0'
)
```

2.2 Map Visualisation (map.html)

```
<!doctype html>
<html>
  <head>
    <title>Choropleth United Kingdom Map - Flu Occurences based on Twitter</title>

    <script src="\static\lib\jquery-1.10.2.min.js"></script>

    <script src="\static\lib\underscore-min.js"></script>
    <script src="\static\lib\d3.v3.min.js"></script>
```

```
    <script src="\static\lib\topojson.v1.min.js"></script>
    <script src="\static\lib\dimple.v1.min.js"></script>
    <script src="\static\lib\numeral.min.js"></script>
    <script src="\static\lib\leaflet.js"></script>

    <script src="\static\mapProcessing.js"></script>
    <script src="\static\choropleth.js"></script>

    <link rel="stylesheet" href="\static\leaflet.css" />
    <link rel="stylesheet" href="\static\styles.css" />
    <link rel="stylesheet" href="\static\main.css" />

  </head>

  <body>

    <div id="container-left">
      <div id="choroplethmap"></div>
    </div>

    <div id="container-right">
        <div id="save-results">Save Results</div> <br/>
        <div id="time-passed">Hours passed : <span id="hours-value">0</span></div>

        <div id="tweet-data"></div>
    </div>

    <script>

      FluUkChoropleth.init('choroplethmap');
      window.hourCounter = 0;

    </script>

    <script>

      var source = new EventSource('/stream');

      source.onmessage = function (event) {

        var county_id = event.data.split("DELIMITER")[0];
        var fluOccurences = event.data.split("DELIMITER")[1];
        var allOccurences = event.data.split("DELIMITER")[2];
        var sentence = event.data.split("DELIMITER")[3];

        console.log("NEW DATA IS HERE " + event.data);

        var county_code = county_id.substr(1, county_id.length);

        console.log("[" + county_code + " ] [ ON TWEET FLU OCC ] : " +
parseInt(fluOccurences));
        console.log("[" + county_code + " ] [ ON TWEET ALLOCC] : " + parseInt(allOccurences));


FluUkChoropleth.updateValues(county_code,parseInt(fluOccurences),parseInt(allOccurences));

      };

    </script>

    <script>

        function download(filename, text) {
            var pom = document.createElement('a');
            pom.setAttribute('href', 'data:text/plain;charset=utf-8,' +
```

```
encodeURIComponent(text));
            pom.setAttribute('download', filename);

            if (document.createEvent) {
                var event = document.createEvent('MouseEvents');
                event.initEvent('click', true, true);
                pom.dispatchEvent(event);
            }
            else {
                pom.click();
            }
        }

        function saveLog() {
            var d = new Date();
            window.hourCounter++;
            $('#hours-value').html(window.hourCounter);
            download('autoLog_hours_passed_'+window.hourCounter+'_date_'+d+'.txt',
FluUkChoropleth.getAllProps());
            console.log("ok autosaved");
        }

        $(document).on( 'click'    , '#save-results', function(e)  {
                var d = new Date();
                console.log("SAVE THE RESULTS");
                download('flu_occurences_uk_'+d+'.txt', FluUkChoropleth.getAllProps());
                console.log("ok saved");
        } );

        var handle = setInterval(saveLog, 3600000);

    </script>

  </body>
</html>
```

2.2.1 Choropleth (choropleth.js)

```
var FluUkChoropleth = {
  init: function(div) {
    this.div = div;
    this.map = L.map(this.div).setView([54.0, -1.5], 6);

   // L.tileLayer('http://{s}.tile.cloudmade.com/{key}/22677/256/{z}/{x}/{y}.png', {
   //   attribution: 'Map data &copy; 2011 OpenStreetMap contributors, Imagery &copy; 2012
CloudMade',
   //   key: 'BC9A493B41014CAABB98F0471D759707'
   // }).addTo(this.map);

var osm = L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
                attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a>
contributors'
          }).addTo(this.map);


    mergedFeatureLayer(this.map, "/static/data/flu_occ_per_code.csv",
                                 "/static/data/topo_lad.json",
                                 "LAD13CD",
                                 this.style,
                                 null,
                                 null,
                                 "lad"
                       );
```

```
  addLegend([0,
             0.0001,
             0.0002,
             0.0003,
             0.0004,
             0.0005,
             0.0006,
             0.0007,
             0.0008], this.map, this.color);

  addInfo(this.map);

},

destroy: function() {
  this.map.remove();
},

refresh: function() {
  this.destroy();
  this.init(this.div);
},

color: function(d) {

  return  d == 'NA' ? 'grey' :
          d == 'undefined' ? '#333333' :
          d >= 0.0008 ? '#c3001d' :
          d >= 0.0007 ? '#E43404' :
          d >= 0.0006 ? '#E48608' :
          d >= 0.0005 ? '#E4D60C' :
          d >= 0.0004 ? '#A4E310' :
          d >= 0.0003 ? '#5AE314' :
          d >= 0.0002 ? '#18E31D' :
          d >= 0.0001 ? '#1CE268' :
          d >= 0.0000 ? '#abe3e3' :
          'grey';

},
style: function(feature) {

  return {
    fillColor: FluUkChoropleth.color(feature.properties.RATE),
    weight: 1,
    opacity: 1,
    color: 'rgba(255,255,255,0.7)',
    fillOpacity: 0.8
  }
},

setDefaultView: function() {

    FluUkChoropleth.map.setView([53.0, -1.5], 5);

},
showAllProps : function() {

    mergedLayer.eachLayer(function (layer) {

        console.log(layer.feature.properties.LAD13CD + " - " +
                    layer.feature.properties.LAD13NM + " -  OCC : " +
                    layer.feature.properties.FLUOCC);

    });
```

```
  },
  getAllProps : function() {

      var res = "";

      mergedLayer.eachLayer(function (layer) {

          res += layer.feature.properties.LAD13CD + "\t" +
                 layer.feature.properties.LAD13NM + "\t" +
                 layer.feature.properties.FLUOCC + "\t" +
                 layer.feature.properties.TOTALOCC + "\n";

      });

      return res;
  },
  setCustomView: function(v,z) {

      FluUkChoropleth.map.setView(v, z);

  },
  updateValues : function(feature, fluOcc, allOcc) {

      mergedLayer.eachLayer(function (layer) {

          if (layer.feature.properties.LAD13CD == feature) {

                layer.feature.properties.FLUOCC = parseInt(fluOcc);
                layer.feature.properties.TOTALOCC = parseInt(allOcc);

                console.log(" [ BEFORE ] FLUOCC : " + layer.feature.properties.FLUOCC +
                          " TOTALOCC : " + layer.feature.properties.TOTALOCC +
                             " RATE : " + layer.feature.properties.RATE +
                             " LAD13CD : " + layer.feature.properties.LAD13CD +
                             "-----------------------------------");

                if (parseInt(allOcc) != 0 ) {
                    layer.feature.properties.RATE = ( parseInt(fluOcc) / parseInt(allOcc) );
                }
                else {
                    layer.feature.properties.RATE = 0.0;
                }

                console.log(" [ AFTER ] FLUOCC : " + layer.feature.properties.FLUOCC +
                             " TOTALOCC : " + layer.feature.properties.TOTALOCC +
                             " RATE : " + layer.feature.properties.RATE +
                             " LAD13CD : " + layer.feature.properties.LAD13CD +
                             "-----------------------------------");

          //    [ BEFORE ] FLUOCC : NaN TOTALOCC : NaN RATE : NaN LAD13CD : E08000018----------
-------------------------
          //    [ AFTER ]  FLUOCC : 0   TOTALOCC : 2   RATE : 0   LAD13CD : E08000018----------
--------------

                layer.setStyle( { fillColor :
FluUkChoropleth.color(layer.feature.properties.RATE) } );
          }

      });

  },
  customStyleOntheFly: function(feature,color) {

    mergedLayer.eachLayer(function (layer) {
```

```
        if (layer.feature.properties.LAD13CD == feature) {

            console.log("THIS : " + layer.feature.properties.LAD13CDO);
            console.log("THIS : " + layer.feature.properties.FLUOCC);

            layer.feature.properties.FLUOCC = parseInt(layer.feature.properties.FLUOCC) + 1;

            console.log("THIS : " + layer.feature.properties.FLUOCC);

            layer.setStyle( { fillColor : color } );

        }
    });

  },

  defaultStyle: function(feature) {
    return {
      outlineColor: "#000000",
      outlineWidth: 0.5,
      weight: 1,
      opacity: 1,
      fillOpacity: 0
    };
  }
}
```

2.2.2 Map Processing (mapProcessing.js)

```
/**
 * Created with IntelliJ IDEA.
 * User: annapawlicka
 * Date: 02/07/2013
 * Time: 14:34
 * Functions to process csv and json data and add info and legend to the map.
 */

var geojson;

var mergedLayer;

var mergedFeatureLayer = function mergedFeatureLayer(map,
                                                     csvDir,
                                                     jsonDir,
                                                     joinFieldKey,
                                                     style,
                                                     onEachFeature,
                                                     pointToLayer,
                                                     featureObject) {

    var buildingData = $.Deferred();

    d3.csv(csvDir, function (csv) {

        if (csv) {
            $.ajax(
                {
                    url: jsonDir,
```

```
                    async: false,
                    data: 'json',

                    success: function (data) {
                        var pcts = topojson.feature(data, data.objects[featureObject])
                        features = pcts.features;
                        data.features = processData(csv, features, joinFieldKey);
                        buildingData.resolve(data);
                    },
                    error: function (xhr, ajaxOptions, thrownError) {
                        console.log(xhr.status + " - " + thrownError);
                    }
                });
        }
        else {
            console.log("Error loading CSV data");
        }
    });

    buildingData.done(function (d) {

        // control that shows state info on hover
                var info = L.control({position: 'topright'});

                info.onAdd = function (map) {
                        this._div = L.DomUtil.create('div', 'info');
                        this.update();
                        return this._div;
                };

                info.update = function (props) {

                        this._div.innerHTML = '<b>United Kingdom Flu Occurency</b> <br/>' +
(props ?
                                '<b>' + props.LAD13NM + '</b> (' + props.LAD13CD + ')<br />' +
                        props.FLUOCC + ' occurences of ' + props.TOTALOCC + ' tweets <br />'+
                '<b>RATE : <b>' + props.RATE

                                : 'Hover over a county');


                };

                info.addTo(map);


        function highlightFeature(e) {
            var layer = e.target;

            layer.setStyle({
                weight: 3,
                color: '#41474e',
                dashArray: '',
                fillOpacity: 0.7
            });


            if (!L.Browser.ie && !L.Browser.opera) {
                layer.bringToFront();
            }

            info.update(layer.feature.properties);

        };
```

```
        function resetHighlight(e) {
                        //mergedLayer.resetStyle(e.target);

            var layer = e.target;

            layer.setStyle({
                weight: 1,
                color: 'white',
                dashArray: '',
                fillOpacity: 0.7
            });

                        info.update();

                };


        function zoomToFeature(e) {
            map.fitBounds(e.target.getBounds());
        };

        function onEachFeature(feature, layer) {
            layer.on({
                mouseover: highlightFeature,
                mouseout: resetHighlight,
                click: zoomToFeature
            });
        };


        mergedLayer = L.geoJson(d, {style: style, onEachFeature: onEachFeature, pointToLayer:
pointToLayer}).addTo(map);
        console.log("Loading merged data: "+csvDir+" and "+jsonDir);
        mergedLayer.bringToFront();

    });
};

var mergedClusteredMarkers = function mergedClusteredMarkers( map,
                                                              csvDir,
                                                              jsonDir,
                                                              joinFieldKey,
                                                              style,
                                                              onEachFeature,
                                                              pointToLayer,
                                                              featureObject,
                                                              iconFeature,
                                                              addPopup,
                                                              getCustomIcon ) {

    var buildingData = $.Deferred();

    d3.csv(csvDir, function (csv) {

        if (csv) {
            $.ajax(
                {
                    url: jsonDir,
                    async: false,
                    data: 'json',

                    success: function (data) {
                        var pcts = topojson.feature(data, data.objects[featureObject])
                        features = pcts.features;
                        data.features = processData(csv, features, joinFieldKey);
```

```
                    buildingData.resolve(data);
                },
                error: function (xhr, ajaxOptions, thrownError) {
                    console.log(xhr.status + " - " + thrownError);
                }
            });
        }
        else {
            console.log("Error loading CSV data");
        }
    });

    buildingData.done(function (d) {
        var markers = new L.MarkerClusterGroup({
            maxClusterRadius: 25,
            disableClusteringAtZoom: 10,
            iconCreateFunction: function (cluster) {
                return L.divIcon({
                    html: '<span style="display:inline-block; vertical-align:middle">'+
cluster.getChildCount()+' </span>',
                    className: 'mycluster',
                    iconSize: null
                });
            },
            spiderfyOnMaxZoom: true,
            showCoverageOnHover: false,
            zoomToBoundsOnClick: true
        });

        for (var i = 0; i < d.features.length; i++) {
            var a = d.features[i].geometry.coordinates;
            var properties = d.features[i].properties;
            var marker = new L.Marker(new L.LatLng(a[1], a[0]), {
                icon: getCustomIcon(properties[iconFeature]) });
            marker.bindPopup(addPopup(properties));
            markers.addLayer(marker);
        }
        markers.addTo(map);
    });
};



function processData(csvData, features, joinKey) {

    var joinFieldObject = {};

    $.each(features, function (index, object) {

        joinFieldObject[joinKey] = object.properties[joinKey];

        var csv_data = _.findWhere(csvData, joinFieldObject);
        $.extend(object.properties, csv_data);
    });
    return features;
};

var featureLayer = function featureLayer(map, jsonDir, defaultStyle, featureObject) {
    var layer = L.geoJson(null, { style: defaultStyle});

    console.log("Loading feature data: "+jsonDir);
    map.addLayer(layer);
    d3.json(jsonDir, function (error, data) {
        var pcts = topojson.feature(data, data.objects[featureObject]);
        var geojsonLayer = L.geoJson(pcts, {style: defaultStyle , onEachFeature:
```

```
onEachFeature}).addTo(map);
        geojsonLayer.bringToBack();
    });

};



function addInfo(map, callback) {

/*  var info = L.control({position: 'bottomright'});

    info.onAdd = function (map) {

        this._div = L.DomUtil.create('div', 'infox');
        this.update();
        return this._div;
    };

    info.update = function (props) {

        if (props) {

            console.log(props);

            this._div.innerHTML = callback(props);
        } else {
            this._div.innerHTML = "Hover over map";
        }
    };

    info.addTo(map);
    map.info = info;*/
};


function addLegend(gradesParam, map, color) {

    var legend = L.control({position: 'bottomleft'});

    legend.onAdd = function (map) {

        this._div = L.DomUtil.create('div', 'info legend'),
            grades = gradesParam,
            labels = [];

        // loop through our density intervals and generate a label with a colored square for
each interval
        for (var i = 0; i < grades.length - 1; i++) {
            this._div.innerHTML +=
                '<i style="background:' + color(grades[i]) + '"></i> ' +
                    grades[i] + ' &ndash; ' + grades[i + 1] + '<br>';
        }
        return this._div;
    };

    legend.addTo(map);
};


function addCategoricalLegend(categories, map, categoricalColor) {
    var legend = L.control({position: 'bottomright'});

    legend.onAdd = function (map) {
```

```
        this._div = L.DomUtil.create('div', 'info legend'),
            grades = categories,
            labels = [];

        // loop through categories and generate a label with a colored square for each category
        for (var i = 0; i < grades.length; i++) {
            this._div.innerHTML +=
                '<i style="background:' + categoricalColor(grades[i]) + '"></i> ' +
                    grades[i] + '<br>';
        }
        return this._div;
    };

    legend.addTo(map);
};

function numberWithCommas(x) {
    return x.toString().replace(/\B(?=(\d{3})+(?!\d))/g, ",");
}

/* Function that is used to estimate the width of the custom tooltip */
function measureText(pText, pFontSize, pStyle) {
    var lDiv = document.createElement('lDiv');

    document.body.appendChild(lDiv);

    if (pStyle != null) {
        lDiv.style = pStyle;
    }
    lDiv.style.fontSize = "" + pFontSize + "px";
    lDiv.style.position = "absolute";
    lDiv.style.left = -1000;
    lDiv.style.top = -1000;

    lDiv.innerHTML = pText;

    var lResult = {
        width: lDiv.clientWidth,
        height: lDiv.clientHeight
    };

    document.body.removeChild(lDiv);
    lDiv = null;

    return lResult;
}
```

2.3 External Javascript Libraries

- d3.v3.min.js
- dimple.v1.min.js
- jquery-1.10.2.min.js
- leaflet.js
- numeral.min.js
- topojson.v1.min.js
- underscore-min.js

3. R code used for Network Analytics

```
##############################
# TIME SERIES CONDITIONNING #
#       VECTORISED          #
##############################

library(snow)
library(snowfall)
sfInit( parallel=TRUE, cpus=4,
        useRscript=TRUE)  ## cpus = depends on your pc
library(date)
library(chron)
library(timeDate)
library(timeSeries)

##############################

#### read data ####

a<-read.table("valuesTimeSlices.csv",header=TRUE,sep=",")

d.o.d.<-as.numeric(dim(a)[1]) # experiment-specfic time-steps

no.of.ts <- 380 # number of Unitary Authorities

### CUSTOM FUNCTIONS TO BE USED ###

t.s.conversion <- function (date.data,stock.data,mode.of.enum,col.ID) {
  if (mode.of.enum == 0 ) {charvec <- (date.data[,1])}
  if (mode.of.enum == 1) {charvec <- (date.data[-d.o.d.,1])}
  charvec <- timeDate(charvec, format = "%Y-%m-%d", FinCenter = "GMT" )
  if (col.ID > 0) {data <- matrix(stock.data[,col.ID])}
  else {data <- matrix(stock.data)}
  TS <- timeSeries(data, charvec)
  return(TS)
}

## CONVERT DATA INTO TIME SERIES ##

tsALL<-as.ts(matrix(data = 0, nrow=d.o.d., ncol=no.of.ts, byrow = FALSE, dimnames = NULL))

for (i in 1:no.of.ts) {
  tsALL[,i] <- t.s.conversion(a,a,0,i+1)
}


##############################
# DIRECTED CROSS CORRELATION #
#        VECTORISED          #
##############################

# function of directed cross correlation

directed.cross.correlation <- function(ts1,ts2,time.concatenated) {
  if (time.concatenated == FALSE) {Dt<-d.o.d.}
  if (time.concatenated == TRUE) {Dt<-d.o.d.-1}
  DCC<-matrix(data = 0, nrow=Dt-1, ncol=1, byrow = FALSE, dimnames = NULL)
  for (tau in 1:Dt-1) {
    DCC[tau]<-(mean(ts1[1:(Dt-tau)]*ts2[(1+tau):Dt])-
                 mean(ts1[1:(Dt-tau)])*mean(ts2[(1+tau):Dt]))/
      (sd(ts1[1:Dt])*sd(ts2[1:Dt]))
  }
```

```
  return(DCC)
}

#### DATA MANIPULATION ####

ADJA <- array(0, dim=c(no.of.ts,no.of.ts,d.o.d.-1))

for (j in 1:no.of.ts) {
  for (k in 1:no.of.ts) {
    if (j != k) {
      ADJA[j,k,]<-directed.cross.correlation(tsALL[,j],tsALL[,k],FALSE)
    }
  }
}

ADJA[is.na(ADJA)] <- 0

#### NETWORK ANALYTICS ####

library(igraph)

# CREATE the adjacency matrix of the graph

A.M.<-graph.adjacency(ADJA[,,30], mode = "directed", weighted = TRUE, diag = FALSE,
                              add.colnames = NULL, add.rownames = NA)


# USE THRESHOLD TO DERIVE SIGNIFICANT LINKS ONLY #


A0.1<-matrix(data = 0, nrow=no.of.ts, ncol=no.of.ts, byrow = FALSE, dimnames = NULL)
  for (j in 1:no.of.ts) {
    for (k in 1:no.of.ts) {
      if (j != k) {
        if (abs(ADJA[j,k,30]) < 1) {
          A0.1[j,k] <-0
        } else {
          A0.1[j,k] <-1
        }
      }
    }
}


A.M.0.1<-graph.adjacency(A0.1, mode = "directed", weighted = TRUE, diag = FALSE,
                                 add.colnames = NULL, add.rownames = NA)


######## GET EDGELIST

h1<-get.edgelist(A.M.0.1)
write.csv(h1, file = "flulinks.csv")


###### PLOT THE GRAPH #####


### NODES ###

# load vertex attributes  ###   with numbers and names
vattr<-read.table("U.A.names.txt", header=TRUE)
vattributes<-data.frame(name1=vattr[,1],name2=vattr[,2])

### LINKS ###
```

```
# load edge attributes  #####

eattr<-read.table("flulinks.txt", header=TRUE)
eattributes<-data.frame(node1= eattr[,1], node2= eattr[,2])

g<-(graph.data.frame)(eattributes, directed= TRUE, vertices= vattributes)

# give the names to the vector vlab

vlab<-V(g)$name2

# PLOT
g<-set.vertex.attribute( g, "label.name",
                         value= vlab)
V(g)$degree <- degree(g, mode = "out")
plot.igraph(g, layout=layout.fruchterman.reingold,
            asp=0.4, margin=0.0001,
            edge.color="red", edge.width=0.05,
            vertex.shape="circle",
            vertex.frame.color="black",
            vertex.color="gold",
            vertex.size= 2,  vertex.label= vlab, vertex.label.cex=0.5)
```

END OF APPENDIX