Homework 4

Due: December 1, 2025 **Points:** 100

In the given examples, what you type is in red and the program prints what is in black. The symbol ↵ means a carriage return (return, enter).

Your program output should look like the output in the examples, except that what you type won't be in red.

1. (*75 points*) This exercise has you query the PubMed database for a list of publications related to a keyword. Although we won't do it here, the list of publication numbers you get back can then be turned into a list of papers with a second query to the PubMed database.
   To access the PubMed database, go to the URL below, replacing `keyword` with the keyword you want to search for, and `num` the number of publications you would like returned:

   ```
   https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?
        db=pubmed&retmode=json&retmax=num&sort=relevance&term=keyword
   ```

   with no spaces and all on a single line.
   So, for example, to find the 20 publications most relevant to "fever", the URL would be:

   ```
   https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?
        db=pubmed&retmode=json&retmax=20&sort=relevance&term=fever
   ```

   with no spaces and all on a single line.
   When you read the contents of this web page, it is in the JSON format. You can turn this into a dictionary easily using the module `json`. The method `json.loads(contents)`, where `contents` is the contents of the web page, returns a dictionary with one entry, the key of which is "esearchresults". The associated value is another dictionary. The part you want is a list of the publication numbers. The key is "idlist" and the value is a list of the numbers.
   You are to print the numbers of that list on a single line, with commas between them (no spaces). So, for the above, your output would look like this:

   ```
   5822579,26772198,20660880,9208885,24176478,27209095,8698996,10913413,24176479,16895496,
   24176472,2200377,29940346,8272282,7567198,7432877,26514056,3056881,23160839,19578318
   ```

   but all on a single line. Note your numbers might differ from these because more relevant publications may be found.
   Please prompt the user for the keyword to search for and the number of references. If the number of references entered is not a positive integer, give an error and quit.
   Examples:

   ```
   Enter a comma-separated list of keywords (no spaces!): covid
   How many references would you like (positive integers only): 7
   33053381,33024307,35105985,35163638,33305456,33617700,34241776
   ```

   ```
   Enter a comma-separated list of keywords (no spaces!): parkinson
   How many references would you like (positive integers only): 10
   30890425,18344392,34034347,16361025,25611507,33237660,32273329,38844056,30149463,30573414
   ```

   To turn in: Please call your program *pubmed.py* and submit it to Canvas
   Hint: The URL begins with "https:", so you will need to use the SSL stub in the sample programs (see *geturl-ssl.py*, for example).

2. (*25 points*) The *birthday problem* asks how many people must be in a room so that the probability of two of them having the same birthday is a given value. This problem has you explore it by simulation. Basically, you will create a series of lists of random numbers of length $n = 2, \ldots,$ and look for duplicates. You will do this 5000 times for each length. For each length, count the number of lists with at least 1 duplicate number; then divide that number

by 5000. That is the (simulated) probability that a list of *n* generated numbers has at least one duplicate. As the random numbers you generate are between 1 and 365 (each one corresponding to a day of the year), this simulates the birthday problem.
Now, breathe deeply and calm down. We will do this in steps!

(a) First, deal with one set of birthdays. Write a function called `onetest(`*`count`*`)` that generates a list of *`count`* random integers between 1 and 365 inclusive, and returns `True` if it contains a duplicate element, and `False` if it does not. You can use one of the functions in *hasdups.py* to test for duplicates, or you can write your own..

(b) Now for the probability for *count* people. Write a function `probab(`*`count, num`*`)` that runs *num* tests of *count* people, and counts the number of tests with duplicates. It returns the fraction of the tests with duplicates; that is, the number of duplicates divided by *num*.

(c) Now for the demonstration. Prompt the user for a probability; we'l luse 0.5 here. Start with 2 people, and begin adding people until the probability of that many people having two people with a birthday in common is over 0.5. (In other words, start with a list of 2 elements, and increase the number of elements in the list until the simulation shows a probability of 0.5 that a number in the list is duplicated.) Then print the number of people. Be sure to check the input for validity. Specifically, the input must be a floating point number between 0 and 1 inclusive. If it is not a number, print "Invalid input; " followed by the system error message. If it is a number but is not between 0 and 1, print "Probability must be between 0 and 1 inclusive". In both cases, quit.

Examples:

```
Probability: 0.25
You need 15 people for two of them to have the same birthday with probability at least 0.25000

Probability: 0.7
You need 30 people for two of them to have the same birthday with probability at least 0.70000

Probability: 1.5
Probability must be between 0 and 1 inclusive

Probability: hello
Invalid input; could not convert string to float: 'hello'
```

To turn in: Please call your program *bday.py* and submit it to Canvas
Hint: Don't be surprised if your numbers are slightly different than the ones shown in the sample output. As randomness is involved, it is possible your numbers will not match the ones shown here.

To turn in: Please call your program *bday.py* and submit it to Canvas