Homework 2

1. Write a Bozosort implementation in Python or Java. Perform some empirical runtime studies on your method. That is, for list sizes of 2, 3, 4, 5, 6, ... (to as high as you want to go), give the average runtime of your method over several trial runs. Present your results in a nicely formatted table.

   My code is included as bozo.py. The results of the tests are included as results.txt. I ran the bozosort 20 times for each array, then I averaged the times for each. The following is these averages (individual times are omitted for lack of space).

n	Average time (ms)
3	0.0
4	0.100004673004
5	0.249993801117
6	2.2500038147
7	14.7000074387
8	70.7499980927
9	761.450004578
10	2437.15000153

   So, as you can see, this algorithm sucks.

2. Implement the Autokey Vigenere cipher, from scratch, in Java, JavaScript, or Python. Treat characters as codepoints.

   See vigenere.pl

3. The following ciphertext was intercepted. You know the message is in English and that the sender used a monoalphabetic substitution cipher. What is the plaintext?

   ```
   RYW QVKOVWPP KT KLV FVBP, LQKU DYZIY FEE WEPW IYZWTEG HWQWUHP, ZP FP
   DWEE AUKDU RK RYW QLXEZI FP RK BGPWET, FUH ZR ZP, Z RVLPR, VWFPKUFXEG
   PFRZPTFIRKVG FUH WUIKLVFOZUO RK FEE. DZRY YZOY YKQW TKV RYW TLRLVW,
   UK QVWHZIRZKU ZU VWOFVH RK ZR ZP JWURLVWH.
   ```

   ```
   THE PROGRESS OF OUR ARMS, UPON WHICH ALL ELSE CHIEFLY DEPENDS, IS AS
   WELL KNOWN TO THE PUBLIC AS TO MYSELF, AND IT IS, I TRUST, REASONABLY
   SATISFACTORY AND ENCOURAGING TO ALL. WITH HIGH HOPE FOR THE FUTURE,
   NO PREDICTION IN REGARD TO IT IS VENTURED.
   ```

   I began by noting that, three quarters of the way through line 2, there is a single 'Z'. The only two words in the English language that are one letter long are 'A' and 'I'. Many of the two-letter words also contained 'Z's. I began assuming 'Z' was 'A'. I then noticed that the most frequent three-letter word was 'RYW', so I replaced 'RYW' with 'THE', respectively. This was enough to develop a lot of the two letter words, but I found that, 'I' worked much better for 'Z' than 'A', and 'A' would work really nicely in certain areas (mostly as other two and three letter words) as 'A'. Another inside was that the second word ends in 'PP'. Words ending in two of the same letter is fairly rare, unless that letter is ‘S’ or ‘E’. ‘E’ was already accounted for so I set ‘P’ to ‘S’. At this point, the first word of the third line began to look very similar to ‘SATISFACTORY’, and once I filled that in the rest of the puzzle started to fall into place. Certain words with distinctive spellings like ‘MYSELF’ and ‘PUBLIC’ helped get fringe letters like ‘P’ and ‘M’, and I realized that the other most common three letter word, ‘FUH’, was almost certainly ‘AND’. At this point the puzzle pretty much just solved itself. It ended up being Lincoln; I was kind of hoping for Zodiac Killer.

4. Decrypt the following ciphertext, given that you know it was encrypted with the bifid algorithm in which the Polybius square was laid out in the usual fashion using the keyphrase “Darn, not another cryptanalysis question”.

```
TWBTLLAEPODTUBTWBTLTDLDDVSNNHEETLSKDDSIFGIIMWLYDKDDSPHBPQKOFHMDLSKRS
```

Our Polybius square is as follows:

	1	2	3	4	5
1	D	A	R	N	O
2	T	H	E	C	Y
3	P	L	S	I	Q
4	U	B	F	G	K
5	M	V	W	X	Z

So it follows that

```
T   W   B   T   L   L   A   E   P   O   D   T   U   B   T   W   B
21  53  42  21  32  32  12  23  31  15  11  21  41 42   21  53  42
T   L   T   D   L   D   D   V   S   N   N   H   E   E   T   L   S
21  32  21  11  32  11  11  52  33  14  14  22  23  23  21  32  33
K   D   D   S   I   F   G   I   I   M   W   L   Y   D   K   D   D
45  11  11  33  34  43  44  34  34  51  53  32  25  11  45  11  11
S   P   H   B   P   Q   K   O   F   H   M   D   L   S   K   R   S
33  31  22  42  31  35  45  15  43  22  51  11  32  33  45  13  33
```

```
21534221323212233115112141422153422132211132111152331414222323213233
45111133344344343451533225114511113331224231354515432251113233451333
COMPUTERSCIENCEISNOMOREABOUTCOMPUTERSTHANASTRONOMYISABOUTTELESCOPESS
```

5. What are the RSA’s public and private keys generated from

$$p = 23847623789462398745236743254827634647$$

$$q = 80147623789462398745236743254827634711$$

My public key can be any number that is coprime with $\phi(n)$. Any prime that does not divide $\phi(n)$ will do. $\phi(n) = (p-1)(q-1)$, so we need a public key e that does not divide $p-1$ or $q-1$. Using Mathematica I find

$$e = 10633823966279326983230456482242756601$$

Using Mathematica again to solve $65537d \equiv 1 \mod \phi(n)$ I get

$d = 2211643320655810747687750830433962306329208078513492712729788036900179852 41$

6. If someone's RSA public key is (729880581317, 5), what is her private key? Give a detailed derivation, showing all work.

   We know that $n = 729880581317 = pq$, but because this is not a secure key length, Mathematica had no trouble finding that the prime factors of n are 822893 and 886969, so those are our p and q (note that p and q are interchangeable in RSA). Thus, $\phi(n) = (p-1)(q-1) = 729878871456$. The public key is 5, which is also a really poor choice, so we know that the modular inverse of 5 and 729878871456 is our private key, and that happens to be 583903097165. So the private key is 583903097165.

7. Dasgupta 1.45 RSA and digital signatures. Recall that in the RSA public-key cryptosystem, each user has a public key $P = (N, e)$ and a secret key d. In a digital signature scheme, there are two algorithms, *sign* and *verify*. The *sign* procedure takes a message and a secret key, then outputs a signature σ. The verify procedure takes a public key (N, e), a signature σ, and a message M, then returns "true" if σ could have been created by *sign* (when called with message M and the secret key corresponding to the public key (N, e)); "false" otherwise.

   (a) Why would we want digital signatures?

   With digital signatures we can confirm that someone we are communicating with is who they say they are. This is important for things like SSL and online encrypted communication; if I'm getting a message from some computer I want to know it's a computer I trust and it's the computer it claims to be, especially if it is asking for privileged information in return.

   (b) An RSA signature consists of $sign(M, d) = M^d \mod N$, where d is a secret key and N is part of the public key. Show that anyone who knows the public key (N, e) can perform $verify((N, e), M^d, M)$, i.e., they can check that a signature really was created by the private key. Give an implementation and prove its correctness.

   Suppose we get a signature σ with public key (N, e) and message M. We know from RSA that $(M^d)^e \equiv M \mod N$. We know that $\sigma \equiv M^d \mod N$. Thus, if we find $\sigma^e \mod N$, it must be congruent to $M \mod N$. If $\sigma^e \not\equiv M \mod N$, then σ was not produced with M or d, meaning this person is a fraud.

   Verify is a one-line Mathematica function:

   ```
   Verify[N_, e_, sigma_, M_] := M == PowerMod[sigma, e, N]
   ```

   I included a file "verify.nb" that illustrates a variety of tests. I'm afraid, other than testing it, I cannot "prove" it works, only illustrate that it does for these tests.

(c) Generate your own RSA modulus $N = pq$, public key e, and private key d (you don't need to use a computer). Pick p and q so you have a 4-digit modulus and work by hand. Now sign your name using the private exponent of this RSA modulus. To do this you will need to specify some one-to-one mapping from strings to integers in $[0, N-1]$. Specify any mapping you like. Give the mapping from your name to numbers $m_1, m_2, ...m_k$, then sign the first number by giving the value $m_1^d \mod N$, and finally show that $(m_1^d)e = m_1 \mod N$.

I chose $p = 113$ and $q = 11$, because these seemed like nice primes that would be easy to work with. I got $N = 1243$. I chose $e = 3$ because it was easy and happened to be coprime with N, then I computed $d = 747$ by realizing that $1243 \times 2 + 1 = 747e$. I used standard ASCII for my name, so $D = 100$. You'd think raising 100 to a power would be easy, but not when it's 747, so I let Mathematica handle that for me and I got 375. I then verified it and it came out as 100, just as expected!

(d) Alice wants to write a message that looks like it was digitally signed by Bob. She notices that Bob's public RSA key is $(17, 391)$. To what exponent should she raise her message?

Since p and q must be greater than 1, an N of 17 is impossible. I will assume the writer accidentally swapped N and e. In this case, 391 factors into 197 and 2, giving us p and q. This makes $\phi(n) = (197-1)(2-1) = 196$. This means that $17d \equiv 1 \mod 196$. Well, using Mathematica I get that $d = 173$. So that is what Alice should use.

8. Dasgupta 1.46 Digital signatures,continued. Consider the signature scheme of Exercise 1.45.

(a) Signing involves decryption, and is therefore risky. Show that if Bob agrees to sign anything he is asked to, Eve can take advantage of this and decrypt any message sent by Alice to Bob.

In order to decrypt a message sent by Alice to Bob, Eve would need Bob's private key. However, if Eve asks Bob for his signature using a message she knows, call it M, Eve will receive $\sigma = M^d \mod N$ back as the signature. Eve can then perform the following algorithm: begin with $x = \sigma$. If M divides x evenly, divide x by M. Otherwise add N to x. Repeat this until x is 1. The number of times you divided by M is d.

(b) Suppose that Bob is more careful, and refuses to sign messages if their signatures look suspiciously like text. (We assume that a randomly chosen message–that is, a random number in the range $\{1, ..., N-1\}$–is very unlikely to look like text.) Describe a way in which Eve can nevertheless still decrypt messages from Alice to Bob, by getting Bob to sign messages whose signatures look random.

The above algorithm works regardless of the message. If you are allowed to supply the message, as long as you keep track of it you can use it.

I thought this was both too easy and too complicated, so I cheated and looked at the answers. The answer for both these is much smarter than what I did. Sending Bob Alice's encrypted message and asking him to sign it will literally decipher it, but then Bob would surely realize what Eve was up to and she would get caught as a bad guy.

9. Dasgupta Problem 2.12

   How many lines, as a function of n (in $\Theta()$ form), does the following program print? Write a recurrence and solve it. You may assume n is a power of 2.

```
function f(n)
    if n > 1:
        print_line(''still going'')
        f(n/2)
        f(n/2)
```

   This problem is a recurrence relation like those on page 53. Here, $a = 2$, $b = 2$, and $d = 0$ since the combining of the answers (printing) takes constant time. By the Master Theorem on page 54, since $\log_2 2 = 1$, it follows that $T(n) = \Theta(n^{\log_2 2}) = \Theta(n)$. This kind of makes sense because we are recursing $\log_2 n$ times, but each time doing double the work, and $2^{\log_2 n} = n$.

10. Dasgupta Problem 2.23 An array $A[1...n]$ is said to have a *majority element if more than half of its entries are the same. given an array, the task is to design an efficient algorithm to tell whether the array has a majority element, and, if so, to find that element. The elements of the array are not necessarily from some ordered domain like the integers, and so there can be no comparisons of the form "is $A[i] > A[j]$?". (Think of the array elements as GIF files, say.) However you* can answer questions of the form: "is $A[i] = A[j]$?" in constant time.

    (a) Show how to solve this problem in $O(n \log n)$ time. (*Hint:* Split the array A into two arrays A_1 and A_2 of half the size. Does knowing the majority elements of A_1 and A_2 help you figure out the majority element of A? If so, you can use a divide-and-conquer approach.)

        Key insight: If A_1 and A_2 both have a majority element, and they are the same element, then that must be the majority element of A as well. If some value x is the majority value of A_1, then half or more of the values in A_1 are equal to x. If the same can be said about A_2, then the "at least one half" of each becomes an "at least one quarter". Well, "at least one quarter" plus "at least one quarter" is "at least one half".

    (b) Can you give a linear-time algorithm? (*Hint:* Here's another divide-and-conquer approach:

        - Pair up the elements of A arbitrarily, to get $n/2$ pairs
        - Look at each pair: if the two elements are different, discard both of them; if they are the same, keep one of them.

Show that after this procedure there are at most $n/2$ elements left, and that they have a majority element if and only if A does.)