Homework Assignment # 1

Due Date: Wednesday, September 13

This homework assignment contains programming problems dealing with characters. In the problems we treat all letters as uppercase letters and ignore periods, commas, etc., and, in particular, blanks between letters. For instance,

`Worcester Polytechnic Institute`

would be written as

`WORCESTERPOLYTECHNICINSTITUTE`.

We consider always the following integer representation of characters: $A \leftrightarrow 0$, $B \leftrightarrow 1$, $\cdots$, $Z \leftrightarrow 25$. The C programming language represents internally character symbols also as integers (lucky for us) using the ASCII code. The ASCII code has the following mapping: $A \leftrightarrow 65$, $B \leftrightarrow 66$, $\cdots$. For our purpose we have thus only to subtract 65 from the character variable in order to get the desired representation for our cipher algorithms, and add 65 to get back to the ASCII representation. It might be a good idea to write the following three (very short) C functions for this problem. Use of the functions, however, is not a requirement.

```
int modulo(int a, int m); /* returns a mod m */
int char2int(char c); /* returns ASCII value of c - 65 */
char int2char(int a); /* returns ASCII character of a + 65 */
```

Problems

0. Read the article "Network Security: It's Time to Take It Seriously," by P. Dowd and J. McHenry, IEEE Computer, September 1998. Hard copies are distributed in class. Please note that the contents of all reading assignment is relevant for the exams.

1. We received the following cipher text encoded with a shift cipher.

   `XULTPAAJCXITLTLXAARPJHTIWTGXKTGHIDHIPXCIWTVGTPILPITGHLXIWIWTXGQADDS`.

   This message can be found on the course web page.

   An elegant method for breaking a shift cipher is letter frequency analysis. Read the subsection "Monoalphabetic Ciphers" in the text book, page 31–33 (note that the monoalphabetic cipher is a more general method than the shift cipher).

   (a) Perform an attack against the cipher based on a letter frequency count: How many letters to you have to identify through a frequency count for recovering the key? What is the cleartext?

   (b) What kind of attack according to our classification was performed?

   (c) Who wrote this message (1 extra point)?

2. (a) Write a C program which decodes the following text:

   `FALSZZTYSYJZYJKYWJRZTYJZTYYNARYJKYSWARZTYEGYYJ`

   using the affine cipher with the parameters $a = 7$, $b = 22$. The message is also on the web page.

   (b) Write a C program which encodes the following text:

   ``` ``Nonsense!'' cried Alice. ```

   using the affine cipher with the parameters $a = 7$, $b = 22$. Remember to convert all characters to uppercase and to ignore blanks and special signs.

3. Find the additive inverse of all elements in Z_m for

   (a) $m = 6$,

   (b) $m = 11$.

4. (a) Which elements in Z_6, Z_7 and Z_9 do not have multiplicative inverses?

   (b) Why is Z_7 different from the two rings Z_6 and Z_9?

   (c) What are the multiplicative inverses of the other elements (you may want to use trial and error for finding the inverses)?

5. What is the multiplicative inverse of 5 in Z_{11}, Z_{12}, and Z_{13}?

   With this simple problem we want now to stress the fact that the inverse of an integer in a given ring depends completely on the ring considered. That is, if the modulus changes, the inverse changes. Hence, it doesn't make sense to talk about an inverse of an element unless it is clear what the modulus is. This fact is crucial for the RSA cryptosystem as we will see in a few weeks.

6. Find all integers n between $0 \leq n < m$ that are relatively prime to m for $m = 4, 5, 9, 26$.

   We denote the *number* of integers n which fulfill the condition by $\phi(m)$, e.g. $\phi(3) = 2$. This function is called "Euler's phi function". What is $\phi(m)$ for $m = 4, 5, 9, 26$?

   (Remark: In the 5th lecture week we will introduce a much faster way of computing $\phi(m)$, given that the factorization of m is known.)

7. What is the key space of the affine cipher?
   ```

8. A popular approach to increasing the security of a private-key algorithm is to apply the same algorithm twice such that:

$$y = e_{k2}(e_{k1}(x))$$

As often in cryptography, things are very tricky and results are often different from the desired ones. In this problem we will show that double encryption of the affine cipher is only as secure as single encryption!

Assume two affine ciphers $e_{k1} = a_1 x + b_1$ and $e_{k2} = a_2 x + b_2$.

(a) Show that there is a single affine cipher $e_{k3} = a_3 x + b_3$ which performs exactly the same encryption (and decryption) as the combination $e_{k2}(e_{k1}(x))$.

(b) Find the values for a_3, b_3 when $a_1 = 3, b_1 = 5$ and $a_2 = 11, b_2 = 7$.

(c) For verification, (i) encrypt the letter `K` first with e_{k1} and the result with e_{k2} and (ii) encrypt the letter `K` with e_{k3}.

(d) Briefly describe what happens if an exhaustive key search attack is applied to a double encrypted affine cipher text. Is the effective key space increased?

Remark: The issue of multiple encryption is of great practical importance in the case of DES, for which multiple encryption most likely does increase security — we will talk about this in a few weeks.