CSCE 230 - Computer Organization

EXERCISE 3: THE CIPHER

Assigned: **Saturday, October 3, 2015**

Due: **Fridays, October 9 and 16**

OBJECTIVES

The objectives of this exercise:

- Computational:
  - Learning about the code generation process for ciphers and encryption programs by creating rules and a one-to-one mapping to encode messages
  - Learning about the trial-and-error code breaking process by trying to guess the mapping to read coded messages
  - Learning about problems caused by ambiguity in the rules used for both mapping and conversion of data into a different representation
  - Learning about the concept of the one-to-one mapping and its significance in problem solving
  - Learning about methodical, algorithmic approaches to solve complex problems such as code breaking
- Creative:
  - Capturing: creating new outputs and using new ways to represent and save data by inventing, testing and documenting a simple system of rules to transform English sentences into a cipher so that their meaning is obscured.
  - Challenging: looking at simple English sentences in new ways (as mysterious encryptions) and considering all the methods you might use to generate or decipher these encryptions.
  - Broadening: acquiring new information and skills by understanding how just three simple rules can generate complex products (ciphers) and how your everyday experience with the English language or with the world can inform your code-generating and code-cracking efforts

READ MORE

PROBLEM DESCRIPTION

For the next two weeks, you will be developing a cipher for the alphabet. What is a cipher exactly? (Check out ***http://en.wikipedia.org/wiki/Cipher*** and

http://en.wikipedia.org/wiki/Morse_code). If you have ever played "Hangman" or watched Wheel of Fortune, that experience will help you in this exercise.

The breaking of such a cipher was one of the first uses for computers. Back in World War II, the Germans had a very advanced (for that time) cipher machine known as Enigma used to relay coded messages to their soldiers and submarines. The British developed a computer called Colossus that used vacuum tubes to perform the large number of calculations necessary to break the Enigma cipher. You are encouraged to explore the technology and design of these early computers. Imagine using relays, tubes, magnetic core memory, etc. to implement the logic that you now recognize as integral to computing. There were teletype machines, magnetic wire/ribbon, and printers.

But ciphers are not the only use of encoding. From our course, we can see encoding being used for convenience, speed, conformity (widely accepted ASCII encoding of characters), and more. Encoding the bits of machine instructions is carefully done to enhance processing speed/space and simplify circuitry in a close dance of cost/benefit.

Your goal is to create a mathematical mapping of the alphabet to encode a message in a way that is not easily decipherable. However, you may create **no more than three** rules for your cipher. These rules are single statements that are used to transform the message. Additionally, you must not change or add words or change word order and all punctuation must be retained unaltered. Basically, all you are allowed to change are the letters.

Here is an example:

Rule 1. The characters are divided into two groups: (1) characters that have an enclosed area in uppercase and (2) characters that do not have an enclosed area in uppercase.

Rule 2. Sort the two groups alphabetically, with group 1 first and then group 2.

Rule 3. Number the sorted list using 1-indexing and use the corresponding number for each character as its code.

READ MORE

Here is the application of the cipher:

Applying Rule 1:

Group 1: { A, B, D, O, P, Q, R } Group 2: { C, E, F, G, H, I, J, K, L, M, N, S, T, U, V, W, X, Y, Z}

Applying Rule 2:

A, B, D, O, P, Q, R, C, E, F, G, H, I, J, K, L, M, N, S, T, U, V, W, X, Y, Z

Applying Rule 3 will produce a 1-to-1 mapping table:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
A	B	D	O	P	Q	R	C	E	F	G	H	I	J	K	L	M	N	S	T	U	V	W	X	Y	Z

This will translate the message:

We drove to the gym and swam in the pool.

To:

23_9|3_7_4_22_9|20_4|20_12_9|11_25_17|1_18_3|19_23_1_17|13_18|20_12_9|5_4_4_16

If you were to try to make each character encode to another randomly selected character, it would require 26 rules, and that would violate the above requirements. That is, you must find a way to treat each character to a transformation using no more than three rules. Please note also that you MUST indicate word breaks (|) and letter breaks (_).

READ MORE

Each group will set up a wiki page on **agora.unl.edu**. The name of this page should be: “Cipher by <Course> Group <Name>” where <Course> is the course abbreviation and <Name> is your group name (e.g., Cipher by CSCE 155A Group Awesome).

Elect one of your group members to login and create this new wiki page. Note that there should be only one page created per group.

1. Week One [20 points]

1.1. Code Generation

Over the course of the first week, you must do the following:

1. Develop the three rules for your cipher. However, keep these rules **internal**, off your wiki. These rules should only be shared amongst your teammates.
2. Post your coded "sentence" of the statement "We drove to the gym and swam in the pool" on your wiki page. Again, please note that you MUST indicate word breaks.
3. Turn in a complete description of the three rules used for your cipher along with a one-to-one mapping of characters to their encoded values to your TA. The one-to-one mapping shows that your code is consistent and two characters can't be mixed up.
4. Then, post three separate coded "sentences" (questions) of your choice on your wiki page as three separate forum discussion threads (scroll down and you will find the link to add new discussion threads). **Each of these sentences must be a question with a readily available answer** so that other teams can attempt to crack your cipher by providing their coded answers to your questions. For example, using our cipher, the question, "What is the capitol of Nebraska?" would be posted as:
   23_12_1_20|13_19|20_12_9|8_1_5_13_20_4_16|4-10|18_9_2_7_1_19_15_1?

   If you then posted as your answer:
   16_13_18_8_4_16_18
   you would be one-third of the way toward breaking the code, with two more questions to answer in code.

The code generation process is still very important in computer science. In the digital age, sensitive data, such as an account password or credit card number, is constantly being transmitted over wireless networks to all computers in range of the access point. To prevent our sensitive data from being stolen, encryption programs have been developed which use *rules* and a *one-to-one* mapping to automatically encode/decode the data. These rules use a series of conditionals (such as if-then-else) and mathematical operations to transform the data and a unique key into unreadable data with a one-to-one mapping. The process is generally done on variables storing fixed "blocks" of data that allows the encryption of everything from text files to movies. The unreadable data is then sent over the network and the process is reversed on the other side to decode the data. This data is worthless unless you already have or can guess the encryption key. Fortunately, using a current supercomputer, trying all possible keys for (current) AES-128 encryption would take a billion billion years (not a typo).

READ MORE

2. Week Two

2.1. Cracking Other Codes [20 points]

This week, your job is to try to crack as many of the other team's codes as possible. There will be 10 bonus points for the team that obtains the longest time duration until its code is cracked and another 10 bonus points if the winner of the second award finishes with its code uncracked.

The code cracking process is also very important in computer science. This process consists of breaking or cracking the encryption so that unreadable, encrypted data can be transformed back into the original data. There are many possible reasons for breaking such encryption ranging from government agencies trying to keep tabs on ne'er-do-wells to hackers trying to steal your credit card information. The most common, brute force, way of cracking the code is to try and guess the encryption key used to encode the data by writing a program which loops through all the possible encryption keys. Such a program could take a very long time to run but, keep in mind, computers are always getting faster. For example, DES 56-bit encryption, which may have been unbreakable years ago, now takes less than ten minutes to brute force. As a result, in any field, those individuals who need to send sensitive data should be aware of the state-of-the-art in encryption.

Our conception of a processor often uses the human brain as an analogy. What we are learning in class is the long-established Von Neumann Architecture. See https://en.wikipedia.org/wiki/Von_Neumann_architecture for an overview, and you will quickly recognize the design. Optimizing this architecture with multi-core, cluster machine, cache memory, pipe-lining and more remains the focus of our courses including CSCE 430 (230 on steroids) and CSCE 432, 456, and 930. Even in our CSCE 230 we will soon be going over chapter 9 which explores ways to dramatically speed up the ALU. Our ability to crack codes keeps improving, but there are proven computational complexity limits. There is hope that a radically different design can break through these limits. Work is slowly progressing on Quantum Computing. See https://en.wikipedia.org/wiki/Quantum_computing for an overview. Manipulating bits in finite state machines (to be introduced this week) is what we spend entire courses covering, beginning with our CSCE 230. Imagine working in the weird world of quantum mechanics with qubits which can be in superpositions of states!

To crack a code, you and your team must go to another team's wiki page, and answer all three of their questions in their code. Post the responses to their questions on their wiki page. It is essential that you post your responses encoded. Otherwise, your responses might make it too

easy for another team to "crack" this code because they would be able to guess at the question based on the answers you gave. Additionally, it shows that you have successfully determined the rules for encoding and decoding.

IMPORTANT: Do not reveal whether another team has cracked your code before the end of the two weeks. You are to do so only after the Week 2 deadline.

READ MORE

2.2. ANALYSIS

Analysis 1 [5 points]: Analyze the questions asked by other teams. Were they too easy to answer? Were the questions faulty?

Analysis 2 [5 points]: How easy would it be to implement your cipher in a computer program? What would your concerns be with implementing your cipher?

2.3. REFLECTION

Reflection 1 [5 points]: How strong are your rules? Are they clear, or is there some level of ambiguity? Could anybody follow them and understand them? How could you improve your rules?

Keep in mind that any ambiguity in the rules would be very bad for encryption used in the real world. Imagine sending a message to a colleague about an important business deal or project only to find out, after the fact, that the original message arrived as unreadable data that could not be decoded. Imagine, further, engineering applications that require mapping or conversion of certain data into a different representation. Two examples are an iPod which coverts music stored in a compressed, digital format into an analogue sound signal and a LED television that converts a cable television signal into pixels on the screen. For both examples, ambiguity in the rules for conversion could result in poor quality media and many unhappy customers.

Reflection 2 [5 points]: Reflect on your "code cracking" approach. What kind of weaknesses and strengths did it have? Imagine you had to decode over 9000 characters rather than just the 26 in the English alphabet. Could you scale your approach? Would brute force work (i.e. simply try to determine a mapping of characters to their cipher)?

DEADLINES AND HAND-IN

Week 1 Deadline – [Friday, October 9, 11:59 p.m.]: You should have a completed set of rules by now, posted the phrase encoded to your wiki page, and **emailed (1) a description of the rules and (2) a one-to-one character mapping to the grading TA**.

Week 2 Deadline – [Friday, October 16, 11:59 p.m.]: Your analyses and reflections are due at the end of this week, as well as your attempts at cracking as many codes as possible. Please post the responses to the analysis and reflection questions on the same wiki you used for week one just below your encode messages. When the week has ended, please post your rules to your wiki page.

END OF EXERCISE
