Programming Assignment #3—Error Detection and Correction

COMP 222 Computer Organization

**Objective:**

To check a Hamming code for a single-bit error, and to report and correct the error (if any).

In the context of this program, **“Hamming Code” refers to the entire word, including all data and check bits.**

**Menu Options:**

The program reads in a Hamming Code which may or may not have a single error. It then determines which bit, if any, is incorrect and corrects it. Finally, when requested by the user, it displays which bit was in error along with the corrected word. If there is not an error, it reports that as well.

***Your Full Name*** (must always appear above the menu)

1) Enter parameters

2) Check Hamming code

4) Quit program

**Inputs:**

* The maximum length of a Hamming code.

This is needed for malloc( ), when allocating space for the input code string, but the actual code entered by the user may have fewer total bits than the maximum. If a code is entered which has more bits than this maximum, your program should report this as an invalid entry.

* The parity of the check bits (even=0, odd=1)
* The Hamming code

**Outputs:**

* The erroneous bit (if any)
* The corrected Hamming code (if there was an error)
* If no error is found, this is also reported.

**Programming Notes:**

* To use the **Math library**, use: “**#include <math.h>”** to access various functions, such as pow(base, exp), log(number), etc.
* If necessary, also include the flag “-lm” when you compile. If you develop your program using MinGW, you do not need to include this option in your command line, BUT eventually you will need it for k200.ecs.csun.edu.

A sample command line to compile your program and utilize the math library is **gcc –lm prgassign3\_yourlastname.c –o prgassign3\_lazik**

* To use the **String library**, use: **“#include <string.h>”** to access various functions, such as strlen(string) which returns an integer representing the length of a string of characters.
* To perform the **XOR** function, use the operator “**^**”.

**Special Programming Hints:**

The following are technical points you may want to consider regarding your solution to this assignment. This does not mean you must use them, but you may.

* The C-language has a **log(x)** function, but it is to the **base-10**. **To convert any Log function from base-a to base-b,** you can use the mathematical relationship:

**Logb(x) = Loga(x) / Loga(b)**

* The C-language has a function that you may want to consider using. It is the **ceil(x)** function which rounds up a double value to the nearest integer value.
* Before exiting your program, don’t forget to free up any memory that was dynamically allocated using “malloc( ).”

**What to turn in:**

* **Softcopy** of **source code** submitted to **http://moodle.csun.edu** via the submission instructions. Be sure to name your source code: ***prgassign3\_yourlastname.c***

Please make sure **all characters in the program’s name are lower case**, including the first letter of your last name.

* **Input and Output formats must exactly adhere to those shown below**.

Any deviation from the sample input/output formats for submission will result in an automatic **10%** reduction in your grade.

* You can use any editor and/or compiler of your choice to do the assignment, but, before you turn in your assignment, **make sure** your code compiles and executes under the **gcc** compiler and the Unix environment on CSUN’s **k200.ecs.csun.edu** server; otherwise you will receive 0 points for compilation and execution.

**For the first two assignments, I permitted students to go back and fix their first two assignments when it was determined that they didn’t compile/execute error free on k200.ecs.csun.edu. The grade, and the grades of all late assignments was reduced by 25% which is better than zero. But I will not be doing this for this assignment and the remaining assignments in the course.**

**IF YOUR SUBMISSION DOES NOT COMPILE/EXECUTE ERROR FREE ON K200.ECS.CSUN.EDU THE FIRST TIME I TRY IT, YOU WILL RECEIVE A GRADE OF ZERO!**

**The very last thing you should do before submitting your program is compile and execute it on k200.ecs.csun.edu using the sample command line above.**

* Hardcopy printed listing of your program. Please place this on the Professor’s desk at the beginning of class on day the assignment is due. Make sure your full name appears on each page of the listing and that all pages are stapled together in their correct order BEFORE you come to class.

**Testing Your Program:**

Use this sample output to verify that your program is running properly.

It is recommended, however, that you also **make up simple test cases of your own, ones you can easily do by hand with a pencil and paper,**  to verify proper operation of your program.

Error detection/correction:

---------------------------

1) Enter parameters

2) Check Hamming code

3) Quit

Enter selection: 1

Enter the maximum length: 12

Enter the parity (0 = even, 1 = odd): 0

Error detection/correction:

---------------------------

1) Enter parameters

2) Check Hamming code

3) Quit

Enter selection: 2

Enter the Hamming code: 100011001010010

**\*\*\* Invalid Entry - Exceeds Maximum Code Length of 12**

Error detection/correction:

---------------------------

1) Enter parameters

2) Check Hamming code

3) Quit

Enter selection: 2

Enter the Hamming code: 1000110

**\*\*\* There is an error in bit: 6**

**\*\*\* The corrected Hamming code is: 1100110**

Error detection/correction:

---------------------------

1) Enter parameters

2) Check Hamming code

3) Quit

Enter selection: 1

Enter the maximum length: 15 (Note change in maximum length.)

Enter the parity (0 = even, 1 = odd): 1 (**Note change in parity!)**

Error detection/correction:

---------------------------

1) Enter parameters

2) Check Hamming code

3) Quit

Enter selection: 2

Enter the Hamming code: 1000110

**\*\*\* There is an error in bit: 1**

**\*\*\* The corrected Hamming code is: 1000111**

Error detection/correction:

---------------------------

1) Enter parameters

2) Check Hamming code

3) Quit

Enter selection: 2

Enter the Hamming code: 1000111

**\*\*\* There is no bit error**

Error detection/correction:

---------------------------

1) Enter parameters

2) Check Hamming code

3) Quit

Enter selection: 3