UMass Boston CS 444
InClass Homework 2
Posted Wednesday, March 25, 2026
Due Saturday, March 28, 2026 in softcopy at 11:59 pm in your course directory

This homework is going to be worked on during the second half of class on Thurs March 26, 2026. If you finish it during class, you may submit it on paper to the instructor. If you wish to work more after class, you may submit it by uploading it to your course directory on the server before 11:59 pm on Thursday.

To submit your homework in softcopy, prepare one or two files. You can prepare and upload a text file named `hw2.txt` – name it exactly `hw2.txt`. Or you can prepare and upload a PDF file called `hw2.pdf` — name it exactly `hw2.pdf`. Or you can submit both, with some work in one and some work in the other.

If you have trouble with uploading, email `operator@cs.umb.edu` for help and copy the instructor.

The questions in this homework are based on the Hamming slides, the readings in Tanenbaum and Bos for Chapters 5 and 6, and the Ch6 Deadlock slides.

1 Hamming Code

1.1

For Hamming(15,11), what is the code for binary 1010 1001 010? What is the code for binary 1010 1101 010?

1.2

What is the Hamming distance between the codes above?

1.3

Copy Hamming slide 6 or 7 but flip bit D4. Be sure you show 2 lines of header and 4 lines of data.

1.4

Explain how you know which bit has the error.

2 Analysis of Figure 6-4. A non-solution fo the Dining Philosopher's problem, p442 in Tanenbaum and Bos and on the class website and passed back today

2.1

Review the code in the figure. Think about the possibilities for whether or not deadlock occurs. Deadlock is quite rare.

2.2

Draw a timeline. (That means draw two axes, vertical for processes and horizontal for time.) Assume all philosophers get hungry at about the same time. Show one horizontal line for each of the five philosophers processes. Use abbreviations for their function calls. Create the timeline of a deadlock.

2.3

Copy your timeline and make one change. Show the philosopher who picked up the last fork as not picking it up. Allow a different philosopher to pick it up and eat. Then get all philosophers back to thinking.

2.4

Draw the circular wait diagrams for the deadlock. Show processes as circles, resources as forks and one-way arrows.

2.5

Explain which arrows mean requested the resource and got it. Label those arrows RRG.

3 Analysis of Figure 6-5, Solution, p443, in Tanenbaum and Bos

3.1

Supposedly, this code cannot deadlock. Setup the timeline the same way as for the non-solution code. Make one change. Remove the mutex that protects the critical region of code. Draw the deadlock that could occur if two philosophers are in the critical region code at the same time and one get swapped out partway though.

4 Virtual Address Space

Instead of a 64-bit machine, consider a 4-bit machine. This means addresses anywhere can only have 4 bits. The largest address is 1111 in binary or F in hex. The total number of addresses is 16, from 0000 to 1111.
Further, consider bytes of memory with addresses. This means the machine is byte addressible. The number of bits in the address limits the number of bytes in memory.
In the example, we show address in hex and in binary. We show data in binary. There is data in only the low 8 bytes to start so the other bytes are random, marked "not set". Here is the example:

```
Address:        Data:
Hex:  Binary: Binary:
F     1111      not set
E     1110      not set
D     1101      not set
C     1100      not set
B     1011      not set
A     1010      not set
9     1001      not set
8     1000      not set
7     0111      11001100
6     0110      00110011
5     0101      11110000
4     0100      00001111
3     0011      10101010
2     0010      01010101
1     0001      11111111
0     0000      00000000
```

4.1

Show the example data moved to start at address 6 hex. What is the address of the byte that is highest in memory now?

4.2

Login to the server and display your `endian.c` file using the command `hexdump -C endian.c` or `hexdump -C <any>.txt`. Select any two lines of what you see as the output. Note that the ASCII is displayed at the right. If there are any characters that are not ASCII, they are displayed as periods. Write the two complete lines exactly as what you see, from the left to the right, including address in hex, data in hex and data in ASCII: