Fall ’18 CIS 314 Assignment 8 – 100/100 points – Due Friday, 11/30, 11:59 PM

Please submit individual source files for coding exercises (see naming conventions below) and a single solution document for non-coding exercises (.txt or .pdf only), when appropriate. Your code and answers need to be documented to the point that the graders can understand your thought process. Full credit will not be awarded if sufficient work is not shown.

Consider the following C code, which is part of a program to simulate a 64B direct-mapped cache with 4B blocks (i.e., 16 cache sets):

```
#include <stdio.h>
#include <stdlib.h>

struct Line {
    unsigned char data[4];
    unsigned int tag;
    unsigned char valid;
};

struct Cache {
    struct Line *lines;
    int numLines;
};

unsigned int getOffset(unsigned int address) {
    // 4B blocks, so offset is bits 0-1
    return address & 0x3;
}

unsigned int getSet(unsigned int address) {
    // 16 sets, so offset is bits 2-6
    return (address >> 2) & 0xF;
}

unsigned int getTag(unsigned int address) {
    // Offset and set are 6 bits total, so tag is high-order 26 bits
    return address >> 6;
}

struct Cache* mallocCache(int numLines) {
    // TODO - malloc a pointer to a struct Cache, malloc a pointer to an array
    // of struct Line instances (array length is numLines).  Also initialize
    // valid to 0 for each struct Line.  Return the struct Cache pointer.
}

void freeCache(struct Cache *cache) {
    free(cache->lines);
    free(cache);
}

void printCache(struct Cache *cache) {
    // TODO - print all valid lines in the cache.
}

void readValue(struct Cache *cache, unsigned int address) {
    // TODO - check the cache for a cached byte at the specified address.
    // If found, indicate a hit and print the byte.  If not found, indicate
    // a miss due to either an invalid line (cold miss) or a tag mismatch
    // (conflict miss).
}

void writeValue(struct Cache *cache, unsigned int address, unsigned char *newData) {
    // Calculate set and tag for address
    unsigned int s = getSet(address);
    unsigned int t = getTag(address);
```

```
    // Get pointer to cache line in the specified set
    struct Line *line = &cache->lines[s];

    // Determine if we have a valid line in the cache that does not contain the
    // specified address - we detect this by checking for a tag mismatch
    if (line->valid && line->tag != t) {
        unsigned char *data = line->data;
        printf("evicting line - set: %x - tag: %x - valid: %u - data: %.2x %.2x %.2x %.2x\n",
                s, line->tag, line->valid, data[0], data[1], data[2], data[3]);
    }

    // Copy new data to line (could use memcpy here instead)
    for (int i = 0; i < 4; ++i) {
        line->data[i] = newData[i];
    }

    // Update line tag, mark line as valid
    line->tag = t;
    line->valid = 1;

    printf("wrote set: %x - tag: %x - valid: %u - data: %.2x %.2x %.2x %.2x\n",
            s, line->tag, line->valid, newData[0], newData[1], newData[2], newData[3]);
}

int main() {
    struct Cache *cache = mallocCache(16);

    // Loop until user enters 'q'
    char c;
    do {
        printf("Enter 'r' for read, 'w' for write, 'p' to print, 'q' to quit: ");
        scanf(" %c", &c);

        if (c == 'r') {
            printf("Enter 32-bit unsigned hex address: ");

            unsigned int a;
            scanf(" %x", &a);

            readValue(cache, a);
        } else if (c == 'w') {
            printf("Enter 32-bit unsigned hex address: ");

            unsigned int a;
            scanf(" %x", &a);

            printf("Enter 32-bit unsigned hex value: ");

            unsigned int v;
            scanf(" %x", &v);

            // Get byte pointer to v
            unsigned char *data = (unsigned char *)&v;

            writeValue(cache, a, data);
        } else if (c == 'p') {
            printCache(cache);
        }
    } while (c != 'q');

    freeCache(cache);
}
```

Copy and paste the above code into a C file and implement the following methods:

1. [30] mallocCache

2. [30] printCache

3. [40] readValue

See inline comments above for descriptions.

Hint: study chapter 6.4 and the writeValue procedure above carefully and ask questions about the parts that you don't understand!

Name your source file 8-1.c.

Here is output from a sample run of the program (your output does not need to match exactly):

```
Enter 'r' for read, 'w' for write, 'p' to print, 'q' to quit: w
Enter 32-bit unsigned hex address: 0x0
Enter 32-bit unsigned hex value: 0xaabb
wrote set: 0 - tag: 0 - valid: 1 - data: bb aa 00 00
Enter 'r' for read, 'w' for write, 'p' to print, 'q' to quit: w
Enter 32-bit unsigned hex address: 0x8
Enter 32-bit unsigned hex value: 0xbbcc
wrote set: 2 - tag: 0 - valid: 1 - data: cc bb 00 00
Enter 'r' for read, 'w' for write, 'p' to print, 'q' to quit: p
set: 0 - tag: 0 - valid: 1 - data: bb aa 00 00
set: 2 - tag: 0 - valid: 1 - data: cc bb 00 00
Enter 'r' for read, 'w' for write, 'p' to print, 'q' to quit: w
Enter 32-bit unsigned hex address: 0x40
Enter 32-bit unsigned hex value: 0xccdd
evicting line - set: 0 - tag: 0 - valid: 1 - data: bb aa 00 00
wrote set: 0 - tag: 1 - valid: 1 - data: dd cc 00 00
Enter 'r' for read, 'w' for write, 'p' to print, 'q' to quit: o
Enter 'r' for read, 'w' for write, 'p' to print, 'q' to quit: p
set: 0 - tag: 1 - valid: 1 - data: dd cc 00 00
set: 2 - tag: 0 - valid: 1 - data: cc bb 00 00
Enter 'r' for read, 'w' for write, 'p' to print, 'q' to quit: r
Enter 32-bit unsigned hex address: 0x0
looking for set: 0 - tag: 0
found set: 0 - tag: 1 - offset: 0 - valid: 1 - data: dd
tags don't match - miss!
Enter 'r' for read, 'w' for write, 'p' to print, 'q' to quit: p
set: 0 - tag: 1 - valid: 1 - data: dd cc 00 00
set: 2 - tag: 0 - valid: 1 - data: cc bb 00 00
Enter 'r' for read, 'w' for write, 'p' to print, 'q' to quit: r
Enter 32-bit unsigned hex address: 0x4
looking for set: 1 - tag: 0
no valid line found - miss!
Enter 'r' for read, 'w' for write, 'p' to print, 'q' to quit: r
Enter 32-bit unsigned hex address: 0x8
looking for set: 2 - tag: 0
found set: 2 - tag: 0 - offset: 0 - valid: 1 - data: cc
hit!
Enter 'r' for read, 'w' for write, 'p' to print, 'q' to quit: r
Enter 32-bit unsigned hex address: 0x40
looking for set: 0 - tag: 1
found set: 0 - tag: 1 - offset: 0 - valid: 1 - data: dd
hit!
Enter 'r' for read, 'w' for write, 'p' to print, 'q' to quit: r
Enter 32-bit unsigned hex address: 0x41
looking for set: 0 - tag: 1
```

```
found set: 0 - tag: 1 - offset: 1 - valid: 1 - data: cc
hit!
Enter 'r' for read, 'w' for write, 'p' to print, 'q' to quit: q
```

Zip the source files and solution document (if applicable), name the .zip file <Your Full Name>Assignment8.zip (e.g., EricWillsAssignment8.zip), and upload the .zip file to Canvas (see Assignments section for submission link).