Instructions: Solve these problems by writing code in a file. It should run and print out the answers with each answer labeled with problem number. As usual, you may collaborate with your classmates and ask for assistance from the TA. But don’t copy anyone else’s answer.

1. Implement each fragment of code, along with a statement to print out in hexadecimal the value in variable z following execution. Be sure to display the right number of bytes for the type of z. If you don’t want to do each in a separate function, you may want to move the variable declarations to the front and/or rename some variables to prevent multiple declaration errors.

(a) int x = 2;
    int y = 4;
    int z = (x << 3) + (y << 2);

(b) int x = 0xabcd;
    int y = 0x1234;
    int z = (~x & y);

(c) int x = 0x0c;
    int y = 0x03;
    int z = (x && y) || (x && !y);

(d) int x = 0x80000000;
    int z = (x >> 4);

(e) unsigned int x = 0x80000000;
    int z = (x >> 4);

(f) char x = 0xa1;
    short z = x;

(g) short x = 0x1234;
    char z = x;

(h) int x = 0x39;
    int *p = &x;
    int z = *p + 2;

(i) unsigned int x = 0x37;
    int z = x;

(j) int x = 0xA7;
    unsigned int z = x;
2. Write a function that takes a hex value representing a single precision floating point number. Return the corresponding float value as a decimal number. You can use a union to make this easy. Test it on hex value: 0xC10C0000.

3. Implement the following functionality in C, supplying an expression to swap the two bytes in the unsigned short. Print the value in hex. Test it on x == 0x1234; it should return 0x3412.

   ```c
   unsigned short x;
   unsigned short swapbytes;
   swapbytes = /* your expression here */;
   ```

4. Write a function that takes a character string value and prints out the bytes stored in memory of its representation (including the terminating 0). Test it on a string containing your full name.