Winter ’18 CIS 314 Assignment 3 – 100/100 points – Due Monday, 2/5, 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.

1. [80] Write a C program with 7 functions:

- (10) struct IntArray* mallocIntArray(int length): allocates, initializes, and returns an instance of the following struct:
  ```
  struct IntArray {
      int length;
      int *dataPtr;
  };
  ```
  Hint: you’ll need two malloc calls – one for the instance and one for the instance’s dataPtr (a pointer to an int array of size *length*).
- (10) void freeIntArray(struct IntArray **arrayPtr): frees and nulls the instance’s dataPtr, zeroes out the length, frees and nulls the instance.
- (10) void readIntArray(struct IntArray *array): prompts and reads ints from the user to fill the array (i.e., read one int for each array index). Your program must not crash if the user enters values that cannot be parsed as ints (we’ll cover this in labs).
- (15) void swap(int *xp, int *yp): swaps the int values stored at the *xp* and *yp* pointers.
- (15) void sortIntArray(struct IntArray *array): sorts the input array using Bubble Sort (Google it) by repeatedly calling your *swap* function as appropriate.
- (10) void printIntArray(struct IntArray *array): prints the array (e.g., “[1, 3, 5, 7]”).
- (10) int main(): read in an int length from the user, call *mallocIntArray* to create an array, call *readIntArray* to prompt the user to fill the array, call *sortIntArray* to sort it, call *printArray* to print the resulting sorted array, then call *freeIntArray* to free the heap memory used by the array.

Name your source file 3-1.c.

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

```
Enter length: 5
Enter int: 3
Enter int: 5
Enter int: 7
Enter int: 8
Enter int: 2
[ 2, 3, 5, 7, 8 ]
```

2. [20] Consider the following C function:

int decode(int x, int y, int z);

When compiled, the resulting IA32 code is:

```
movl  12(%ebp), %edx
subl  16(%ebp), %edx
movl  %edx, %eax
imull 8(%ebp), %edx
sall  $31, %eax
sarl  $31, %eax
xorl  %edx, %eax
```

Parameters *x*, *y*, and *z* are stored at memory locations with offsets 8, 12, and 16, respectively, relative to the address in register %ebp. The return value will be stored in register %eax.

Your task is to reverse engineer the C code; specifically to write a C implementation for the decode function that is functionally equivalent to the compiled IA32 code above.

Here are some test runs:

```
decode(1, 2, 4): -2
decode(-4, -8, -12): -16
```

Also write a main() function to test your function. Name your source file 3-2.c

Zip the source files and solution document (if applicable), name the .zip file <Your Full Name>Assignment3.zip (e.g., EricWillsAssignment3.zip), and upload the .zip file to Canvas (see Assignments section for submission link).