Josh Levy – EECS 349 HW2 part 3

1) This assembly code performs the following operations in C:

```
int main (int argc, char **argv, char **envp) {
   int x = 3, y = 5, z = 0;
   z = (x*y) - (x/2);
   printf("%d", z);
}
```

The code prints 14.

2) This assembly code performs the following operations in C:

```
int main (int argc, char **argv, char **envp) {
        int array[8] = {0xC, 0xF, 0xDD, 3, 0x1B0, 0x36, 0x10, 0x43};
        int max = 0;
        int iter = 0;

        while (iter <= 7) {
                if (array[iter] > sum) {
                        max = array[iter];
                }
                iter += 1;
        }
        printf(“%d”, max);
}
```

The code finds the maximum value in the array and prints it, resulting in 432.

3)

```
int main (int argc, char **argv, char **envp) {
        int var1;
        int var2;
        int iter = 100;
        int acc;
        while (iter <= 999) {
                int temp = (int)((long long)(iter * 0x51EB851F) >> 32)
                acc = (temp >> 5) + (iter < 0 ? 1 : 0)
                temp = (int)((long long)((iter + acc * -100) * 0x66666667) >> 32)
                var1 = (temp >> 2) + ((iter + acc * -100) < 0 ? 1 : 0)
                temp = (int)((long long)(iter * 0x66666667) >>32)
                temp = (temp >> 2) + (iter < 0 ? 1 : 0)
                var2 = iter – ((((temp << 2) + temp) * 2) << 2) * 2
                if (acc * acc * acc + var1 * var1 * var1 == iter) {
                        printf(“%d”, iter);
                }
                iter++
        }
}
```

This code does not look like any algorithm I have seen before. My guess is it is obfuscated, since the value of var2 is never read and there are a series of complex operations that ultimately seems to have little purpose. In fact, when we ran this code it literally did nothing.

I added the temp variable inside the loop to make the C code a bit easier to look at, although I don't believe this variable existed in the original source code.

4)

```
#include <stdio.h>

int proc1(int *array, int length, int arg_8) {
    int v10 = 0, vc = 0, v8, v4 = 0;
    //loc_4015B7
    while (v4 < length) {
        v8 = 1;
        //loc_40155E
        while (v8 < arg_8) {
            //loc_401538
            while (array[vc] == 0) {
                vc = (vc + 1) % length;
            }
            v8 += 1;
            vc = (vc + 1) % length;
        }
        //loc_401575
        while (array[vc] == 0) {
            vc = (vc + 1) % length;
        }
        v10 = array[vc];
        array[vc] = 0;
        v4 += 1;
    }
    return v10;
}

int main (int argc, char **argv, char **envp) {
    int array[99];
    int v1a8 = 7;
    int length = 100;
    int index = 0;
    while (index < length) {
        array[index] = index + 1;
        index += 1;
    }
    printf("%d\n", proc1(array, length, v1a8));
}
```

This code actually prints a value, it prints 50. The code still seems very arbitrary, all it does is iterate through the array that is given to it, setting everything to 0 until either the array runs out of entries or it reaches it’s maximum number of values.