CS220 Spring 2022 Exam 1 **Name:___________________________________**

Answer the following questions and show all work. I can't give partial credit if you get an answer wrong and don't show any work.

78 total possible points. The mean was 61 out of 78 or %79.3. The median was 63.5, and the high grade was a 77 out of 78.

1. C has a data type called a **short** that is a two byte integer on the Raspberry Pi. For example, you can declare a variable to be either a **signed** or **unsigned short**. Give the declarations below …

```
short x;
unsigned short y;
```

Some of you didn't read problem carefully. I even underlined it above. A two byte integer is 16 bits.

a. [3] What is the smallest possible value of **x**, expressed in decimal.

-2^{15} or -32768

b. [3] What is the smallest possible value of **x**, expressed in binary.

A one followed by fifteen zeros, or 1000000000000000.

c. [3] What is the smallest possible value of **x**, expressed in hexadecimal.

0x8000

d. [3] What is the largest possible value of **y** expressed in binary.

y is unsigned, so no leading sign bit, it is all ones, sixteen ones, 1111111111111111, in decimal 65535

e. [3] The **%x** modifier in a **printf** format string will print an integer in hexadecimal. What will the following C statement print? `printf("%x", -1);`

```
ffffffff
```

f. [3] What would be printed by `printf("%d", sizeof(x));`

2

g. [5] What is printed by the following program?

This one is slightly tricky. 90 + 88 is 178, which cannot fit into a signed integer, so the result is negative and z is not greater than x.

```
#include <stdio.h>
int main() {
    signed char x, y, z;
    x = 90;                        Output: Bar
    y = 88;
    z = x + y;
    if (z > x)
        printf("Foo\n");
    else
        printf("Bar\n");
}
```

2. [5] What is the output of the following program? Watchout! This is very similar to a study question. But not identical.

```
#include <stdio.h>
int main() {
    int s = 0;                     Output:4
    int n = 40;
    while (n > 0) {
       s = s + !(n & 1);
       n = n >> 1;
    }
    printf("%d\n", s);
}
```

Instead of counting the ones in the number 40 it counts the zeros. 40 in binary is 101000. There are four zeros, so it prints 4.

3. [2] The **&** operator applied to a variable (as in **&x**) is called the **address-of** operator.

4. [2] Adding two positive integers with the result being negative is called **overflow**.

5. [2] If $\mathbf{2^x = 1024}$ then **x** must be **10 (in base-ten)**.

6. [2] Express **-33** as an 8 bit two's complement integer. Show all work.

Probably the easiest way is to write 33 as an 8 bit number and take the two's complement.

33 = 0b00100001   invert the bits and add one and you get 0b11011111

7. [2] `0xDeafBeef` is a valid C hexadecimal constant. True/False Answer: **True**

8. [5] Write a very short C code fragment that declares a variable **x** to be an integer and **p** to be a pointer to an integer. Have **p** point to the integer **x**.

```
int x;
int *p;
p = &x;
```

9. The formula for converting fahrenheit to celsius is *c = (f-32)5/9*.

a. [5] Make a directory **exam1** in your course repository.

b. [5] In the **exam1** directory, create a header file **f2c.h** that declares a function named **f2c** that takes a double and returns a double.

c. [10] In the **exam1** directory create a file **f2c.c** that implements the function **f2c**.

d. [10] In the **exam1** directory create a file **main.c** that takes a command line argument (the argc, argv stuff) and prints the argument converted to celsius. The function **atof** declared in **stdlib.h** converts a string to a double.

e. [5] Push the files **f2c.h**, **f2c.c**, and **main.c** to your GitHub repository. Log in to GitHub to make sure they are there. Do not modify the files after they are pushed. They are timestamped.

```
// f2c.h
extern double f2c(double f);
```

```
// f2c.c
double f2c(double f) {
    return (f - 32)*5.0/9;  // careful, 5/9 is zero
}
```

```
// main.c
#include "f2c.h"
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[]) {
    printf("%.2f\n", f2c(atof(argv[1])));
}
```