CSC 373 Winter 2020 Prof. Lytinen
Midterm Information

- In class: Thursday, Feb 6, 11:50-1:20
- On line: register on D2L for a 90 minute period; must be proctored

Open book, open notes, coputer use and calculators allowed

Material:

- C, especially data types, pointers, arrays, strings, bitwise operators, and shift operators
- Representation of integers
- Bases: 2, 8, 10, 16
- Representation of floating point numbers
- Simple assembly language: instructions that only use registers

Practice Problems

1. List the steps of the C compilation process.

2. Give the order, from fastest to slowest, of the following kinds of memory:

   a. Disk memory
   b. Registers
   c. Cache memory
   d. Main memory (RAM)

3. Why do computers have different kinds of memory?

4. Consider this program. What is the last line of output that it prints? Explain.

```
int main() {
  short x=1;
  while (x > 0) {
    printf("%x\n", x);
    x *= 2; }
}
```

5. What is the hex representation of the largest int? For the smallest (negative) int? Explain.

6. We would like to write a function called **min_and_max,** which finds the minimum and maximum integers in an array. For example, in the array {3, 1, 2, 3, 6, 2, 8, 0, 0, 0}, the min is 0 and the max is 8.

   a) Write a prototype for this function. Keep in mind that **two** integers must be “returned”
   b) Write the **min_and_max** function.

7. Write the functions below so that the elements in **src** are swapped with the elements in **dest.** Write version 1 using array syntax; and version 2 using pointer syntax. Here are prototypes for the two versions:

```
void swap_v1(int numbers1[], int numbers2[], int len);
void swap_v2(int *numbers1, int *numbers2, int len);
```

8. Write a function **index_of** which returns the index of first occurrence of an integer n in an array of integers x, or -1 if n is not in x. Use array syntax. Consider what the prototype for this function must be. For example:

9. Write a function **cindex_of** which returns the index of the first occurrence of a character c in a string s, or -1 if c is not in the string. Use pointer syntax.

10. Write a function which returns a string containing the first x letters of the alphabet, where x is a positive integer less than or equal to 26.

11. Write a function called **bit_on**. It is passed one parameter x (a char) and returns a char whose nth bit is 1, and whose other bits are not modified. By convention, bit 0 is the rightmost bit in a number, bit 1 is the 2nd from the right, etc. Bit 7 is the leftmost bit in a char. Use only bitwise and shift operators.

    For example:

```
char x = ‘a’;
char y = bit_on(x,4);
printf(“%#x %#x\n”, x, y);
```

    The output is 0x61 0x71

12. Write a function called **bit_off**. It is passed one parameter x (a char) and returns a char whose nth bit is 0, and whose other bits are not modified. By convention, bit 0 is the rightmost bit in a number, bit 1 is the 2nd from the right, etc. Bit 7 is the leftmost bit in a char. Use only bitwise and shift operators.

    For example:

```
char x = ‘a’;
char y = bit_off(x,5);
printf(“%#x %#x\n”, x, y);
```

    The output is 0x61 0x41

14. Fill in the table below. Any binary number that starts with 1 is a negative number in 2s complement. Likewise for any hex number that starts with 8-f. Assume that the numbers are represented in 1 byte.

Decimal	8-bit Binary (2s complement)	2-digit Hex (2s-complement)
22		
-22		
	11110110	
		0x22
		0x92

15. Fill in the table.

Base 10	**Binary floating pt**	**Binary scientific**	**IEEE 32-bit**
2 1/2			
	0.11		
		$1.11 * 2^1$	
			010000001000000…000

16. Write a C function which emulates the behavior of this assembly language function. By this, I mean that if your C function is passed the same parameter(s) as f, it returns the same value. **Your C code does not have to compile to the exact same assembly language code.** Also, write a brief description of what the function does.

```
f:
        cmpl    %esi, %edi
        setl    %cl
        xorl    %eax, %eax
        cmpl    %edx, %esi
        setl    %al
        andl    %ecx, %eax
        ret
```