**CSC 373 Winter 2020 Prof. Lytinen**
**Midterm Practice Sample Solutions**

**Practice Problems**

1. List the steps of the C compilation process.

   Preprocessor -> Compiler -> Assembler -> Linker

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)

   B-C-D-A

3. Why do computers have different kinds of memory?

   Cost vs. speed

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; }
}
```

   Answer:

   **4000**

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

   Largest: 0x7fffffff
   Smallest: 0x80000000

6. We would like to write a function called **min_and_max,** which finds 2 integers from an array whose sum is the largest. For example, in the array {3, 1, 2, 3, 6, 2, 8, 0, 0, 0}, the largest two integers are 8 and 6.

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

```
void min_and_max(int *, int, int *, int *);
```

```
// assume n contains at least 1 integer

void min_and_max(int n[], int len, int *min, int *max) {
   *min = n[0];
   *max = n[0];
   int i;
   for (i=0; i<len; i++) {
      if (n[i] < *min)
         *min = n[i];
      if (n[i] > *max)
         *max = n[i];
   }
}
```

7. swap

```
void swap_v1(int x[], int y[], int len) {
   int i, temp;
   for (i=0; i<len; i++) {
      temp = x[i];
      x[i] = y[i];
      y[i] = temp;
   }
}

void swap_v2(int *x, int *y, int len) {
   int i, temp;
   for (i=0; i<len; i++) {
     temp = *(x+i);
     *(x+i) = *(y+i);
     *(y+i) = temp;
}
```

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:

```
int index_of(int x, int arr[], int len) {
   int i;
   for (i=0; i<len; i++)
         if (x == arr[i])
               return i;
   return -1;
}
```

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

```
int cindex_of(char c, char *s) {
  int i;
  for (i=0; *(s+i) != '\0' && *(s+i) != c; i++);
  if *(s+i) != '\0'
    return i;
  else return -1;
```

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. For example:

```
char *first_x(int x) { //, char buffer[]) {
   int i;
   char c = 'a';
//   char c = 97;
   for (i=0; i<x; i++)
      buffer[i] = c++;
   return buffer;
}
```

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 $2^{nd}$ from the right, etc. Bit 7 is the leftmost bit in a char. Use only bitwise and shift operators.

```
char bit_on(char x, int i) {
  return (1 << i) | x;
}
```

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 $2^{nd}$ from the right, etc. Bit 7 is the leftmost bit in a char. Use only bitwise and shift operators.

```
char bit_off(char x, int i) {
  return x & ~(1 << i);
}
```

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 | **00010110** | **0x16** |
| -22 | **11101110** | **0xea** |
| **-10** | 11110110 | **0xf6** |
| **34** | **00100010** | 0x22 |
| **-110** | **10010010** | 0x92 |

15. Fill in the table.

| **Base 10** | **Binary floating pt** | **Binary scientific** | **IEEE 32-bit** |
|---|---|---|---|
| 2 1/2 | **10.1** | **1.01*2^1** | **0 10000000 01000000000000000000000** |
| **.75** | 0.11 | **1.1 * 2^-1** | **0 01111110 10000000000000000000000** |
| **3.5** | **11.1** | 1.11 * $2^1$ | **0 10000000 11000000000000000000000** |
| **3** | **11.** | **1.1 * 2^1** | 0 10000000 10000000000000000000000 |

16.

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

```
int f(int x, int y, int z) {
        int a;  //%rax
        int c;  // %rcx
        if (x < y)
          c = 1;
        a = 0;
        if (y < z)
          a = 1;
        return a&c;
}
```