1. [Practice Problem 2.43]
   ```c
   #define M /* Mystery number 1 */
   #define N /* Mystery number 2 */
   int arith(int x, int y) {
       int result = x*M + y/N;
       return result;
   }
   ```
   Here is a translation—back into C—of the compiler-optimized code. What are the values of M and N?
   ```c
   int optarith(int x, int y) {
       int t = x;
       x <<= 5;
       x -= t;
       if (y < 0) y += 7;
       y >>= 3;
       return x+y;
   }
   ```

2. [Practice Problem 2.27] Write a function with the following prototype which returns 1 if the arguments x and y can be added without causing overflow.
   ```c
   int uadd_ok(unsigned x, unsigned y);
   ```

3. [Practice Problem 2.31, paraphrased] What is wrong with the following code for a function that is supposed to determine whether the arguments x and y can be added without causing overflow?
   ```c
   int tadd_ok(int x, int y) {
       int sum = x+y;
       return (sum - x == y) && (sum - y == x);
   }
   ```

4. a. On a computer with 32-bit integers and pointers, what is the address, in hex, of the most significant byte of a[47]?
   ```c
   int a[1000];
   printf("%x\n", (unsigned int*) a); // prints bfffe550
   ```
   b. Write an expression of type int whose value is the most significant byte of a[47].
5. [Practice Problem 3.37] Consider the following source code, where M and N are constants declared with `#define`.

```c
int mat1[M][N];
int mat2[N][M];

int sum_element(int i, int j) {
    return mat1[i][j] + mat2[j][i];
}
```

Here is a fragment of the assembly code generated by gcc. What are the values of M and N?

```
# i at %ebp+8, j at %ebp+12
movl 8(%ebp), %ecx
movl 12(%ebp), %edx
leal 0(%ecx,8), %eax
subl %ecx, %eax
addl %edx, %eax
leal (%edx,%edx,4), %edx
addl %ecx, %edx
movl mat1(,%eax,4), %eax
addl mat2(,%edx,4), %eax
```

6. [Practice Problem 8.3, slightly modified] List all possible output sequences for the following program.

```c
int main() {
    int f = fork();

    if (f == 0)
        printf("a");
    else {
        printf("b");
        waitpid(-1, NULL, 0);
    }

    if (f == 0)
        printf("c");
    else
        printf("d");

    exit(0);
}
```