Note: I do not have answers to these questions. Nor do I intend to provide answers to them. You should work them out among yourselves. You should have learned all that is necessary from your project assignments already but if you still have problems with these questions, you may ask me specific questions. I will not answer questions like "how do you do this one?" because I want to educate students and do not want to teach students how to prepare for tests...

Question 1.

Suppose registers d0 and d1, and memory locations 8000, 8001, 8002 and 8003 current have the following bit pattern:

```
    +---+---+---+---+---+---+---+
    |   |   |   |   |   |   |   |
    ---------------+---+---+---+---+---+---+---+
    | 00000001 | 00000001 | 00000001 | 00000001 |
    ---------------+---+---+---+---+---+---+---+
    | 11111111 | 11111111 | 11111111 | 11111111 |
    +---+---+---+---+---+---+---+

D0: 8000 | 10101010 |
+---------+---+---+---+---+---+---+---+
D1: 8001 | 01010101 |
+---------+---+---+---+---+---+---+---+
D0: 8002 | 00000000 |
+---------+---+---+---+---+---+---+---+
D0: 8003 | 11110000 |
+---------+---+---+---+---+---+---+---+
```

1. What is the bit pattern in register d1 after the CPU executes the instruction move.b d0, d1 ?

2. Starting with the original values in the registers and in memory, what is the bit pattern in register d1 after the CPU executes the instruction move.w d0, d1 ?

3. Starting with the original values in the registers and in memory, what is the bit pattern in register d1 after the CPU executes the instruction move.l d0, d1 ?

4. Starting with the original values in the registers and in memory, what is the bit pattern in register d1 after the CPU executes the instruction move 8000, d1 ?

5. Starting with the original values in the registers and in memory, what is the bit pattern in register d1 after the CPU executes the instruction move #75, d1 ?
6. Starting with the original values in the registers and in memory, what is the bit pattern in register d1 after the CPU executes the instruction `move #~75, d1`?

7. Starting with the original values in the registers and in memory, what is the bit pattern in the memory after the CPU executes the instruction `move d0, 8000`?

8. Starting with the original values in the registers and in memory, what is the bit pattern in the memory after the CPU executes the instruction `move #~1, 8000`?

9. Starting with the original values in the registers and in memory, what is the bit pattern in the memory after the CPU executes the instruction `move.l d0, 8000`?

10. Starting with the original values in the registers and in memory, what is the bit pattern in the memory after the CPU executes the instruction `move.l #~1, 8000`?

**Question 2.**

An array of integer *MyIntArr*, an array of short *MyShortArr*, three integer variables *i*, *j*, *k*, *IntVar*, and one short variable *ShortVar* are defined as following in assembler:

```
MyIntArr: ds.l 100
MyShortArr: ds.w 10
i: ds.l 1
j: ds.l 1
k: ds.l 1
IntVar: ds.l 1
ShortVar: ds.w 1
```

Write the M68000 assembler instructions that accomplishes the equivalent of the following high level language statements:

1. `MyIntArr[k] = 1234;`
2. `MyShortArr[i+j] = 1234;`
Question 3.
A class is defined as follows:

```java
class MyObj {
    int SSN;
    int Balance;
    MyObj next;
}
```

Assumed that a linked list of MyObj objects has been previously constructed and the assembler variable

```assembly
head: ds.l 1
```

refers to the first object in the list (i.e., contains the address of the first object in the list).

Write the M68000 assembler instructions that accomplishes the equivalent of the following high level language statements:

1. `head.next.SSN = head.SSN;`
2. `head.Balance = head.next.next.Balance + 1234;`

Question 4.
Given is the following class definition:

```java
class MyObj {
    short A;
    int B;
};
```

An array Arr of 10 MyObj objects, a short variable i and a long variable j are defined as follows in M68000 assembler:

```assembly
Arr: ds.l 60
i:   ds.w 1
j:   ds.l 1
```
Translate the following assigns into M68000 assembler instructions:

- \( \text{Arr[i+j].B = Arr[i+4].B + Arr[j+4].A + 1234;} \)

**Question 5.**

The following is the famous Euler algorithm to determine the Greatest Common Divisor (GCD) of 2 numbers A and B. Translate the following program fragment into M68000 assembler instructions:

```c
int A, B, X, GDC;

if (B > A)
{
  X = A;
  A = B;
  B = X;
}

X = A % B;
while (X != 0)
{
  A = B;
  B = X;
  X = A % B;
}

GDC = B;
```

**Question 5.**

The variables \( x \) and \( y \) are integers. Translate the following program fragment into M68000 assembler instructions:

```c
if ( ( x + y >= 70 ) )
{
  x = x + 1;
}
else if ( x < y )
{
  y = y - 1;
}
```