Instructions:

- Print your name in the space provided below.
- This examination is closed book and closed notes, aside from the permitted one-page formula sheet and the MIPS reference card. No calculators or other computing devices may be used.
- Answer each question in the space provided. If you need to continue an answer onto the back of a page, clearly indicate that and label the continuation with the question number.
- If you want partial credit, justify your answers, even when justification is not explicitly required.
- There are 6 questions, priced as marked. The maximum score is 100.
- When you have completed the test, sign the pledge at the bottom of this page and turn in the test.
- Note that either failing to return this test, or discussing its content with a student who has not taken it is a violation of the Honor Code.

Do not start the test until instructed to do so!

Name _____
printed

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

_____
signed

1. Consider the following integer X, represented in 32-bit 2's complement form:

```
0000 1001 0000 1001 0000 1001 0000 1001
```

a) [10 points] What is the representation for –X?

Flip the bits and add 1: `1111 0110 1111 0110 1111 0110 1111 0111`

b) [10 points] Why is 2's complement representation used for signed integers instead of sign-magnitude representation? Be precise.

If sign-magnitude representation is used, then addition of integers with opposite signs requires a different algorithm than addition of integers with the same sign. That would require different addition hardware for the two cases.

If 2's complement notation is used, then the same addition algorithm works in all cases.

Note: the essential point is the addition with mixed signs does not correctly work using the usual add/carry algorithm. With same-signed operands, it's fine. Increased range of representation and having a single representation for zero are only minor advantages of 2's complement representation.

---

2. [10 points] Given two 32-bit signed integers, A and B, if the difference A – B is computed, is it possible that *overflow* will occur? Define *overflow* and justify your conclusion.

Overflow occurs when the mathematically-correct result of a computation cannot be represented correctly in the number of bits (and representation scheme) that is being used.

In this case overflow could occur in the case where A = 0 and B = INT_MIN. Since INT_MIN would be -2^{31}. But then A – B would equal 2^{31} and that is not representable in 2's complement with 32 bits.

And, there are other obvious cases…

Note: a complete answer here needed an example illustrating that this was, in fact, possible.

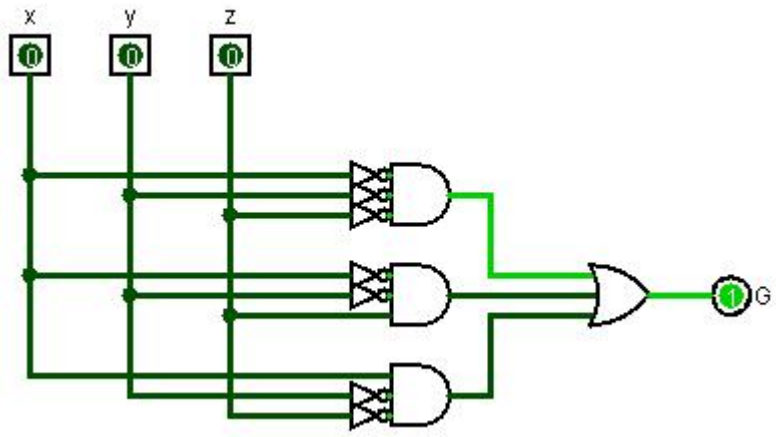
3. The following table defines a Boolean function G:

x	y	z	G
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

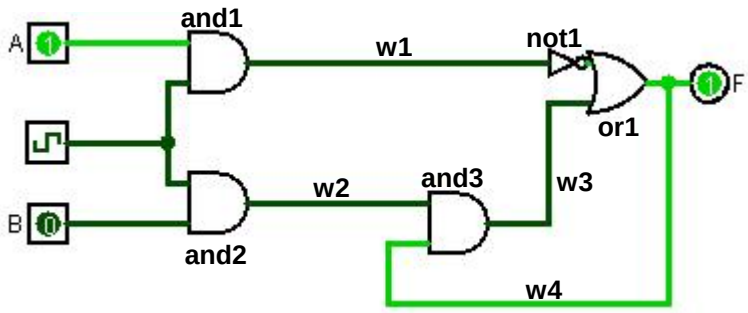
a) [15 points] Write the sum-of-products formula for the function G.

$$G = \bar{x} \cdot \bar{y} \cdot \bar{z} + \bar{x} \cdot \bar{y} \cdot z + x \cdot \bar{y} \cdot \bar{z}$$

b) [10 points] Draw the corresponding circuit to compute G.



4. Consider the circuit shown below, with input A set to 1 as shown:



a) [10 points] Describe in detail how the output value F will change as the clock cycles from low to high:

When the clock goes high, and1 will emit a 1 on w1, which will cause not1 to emit a 0, which will cause the output of or1 to change to 0.

and2 will still emit a 0 on w2, and so nothing else will change in the circuit.

Note: a complete answer needed to either address the data flow in the lower part of the circuit or else point out explicitly that since or1 is already receiving a 1 as its upper input the lower input does not matter.

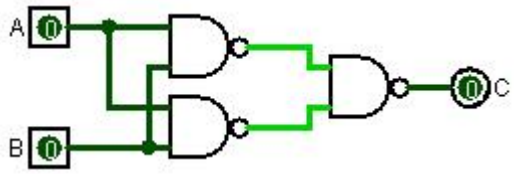
b) [10 points] Describe in detail how the output value F will change as the clock cycles from high back to low:

When the clock goes back to low, and1 will emit a 0 on w1, which will cause not1 to emit a 1, causing the output of or1 to change back to 1.

and2 will still emit a 0 on w2, and so nothing else will change in the circuit.

5. [10 points] Give the simplest explanation of what each circuit shown below actually computes. Justify your conclusions.

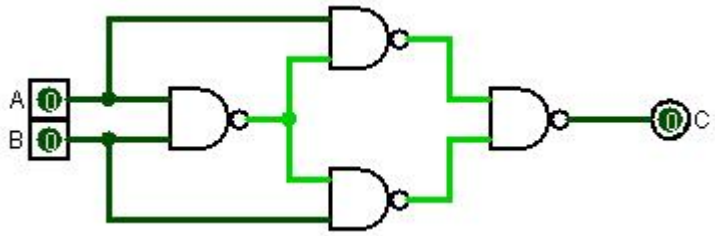
a)




```
C = ~( ~(A * B) * ~(A * B) )
  = ~~(A * B) + ~~(A * B)
  = (A * B) + (A * B)
  = A * B
```

So, this circuit computes the AND function.

b)




```
C = ~( ~(A * ~(A * B) ) * ~( ~(A * B) * B) )
  = ~~(A * ~(A * B) ) + ~~( ~(A * B) * B)
  = (A * ~(A * B) ) + ( ~(A * B) * B)
  = ~(A * B) * (A + B)
  = (~A + ~B)*(A + B)
  = ~A * A + ~A * B + ~B * A + ~B * B
  = ~A * B + A * ~B
```

So, this circuit computes the XOR function.

6. [15 points] Use the algebra of Boolean expressions to prove the Boolean equation given below is correct. You may use any of the rules given on the supplement to this test. If you want to use any other rules, you must prove them as well. You must justify each step by referring to the rule used; it's acceptable to combine steps but you must list every rule that is used.

$$a + (\overline{a} \cdot b) = a + b$$

$a + (\overline{a} \cdot b) = (a + \overline{a}) \cdot (a + b)$ Distributive Law

$= 1 \cdot (a + b)$ Law of Complements

$= a + b$ Boundedness Law