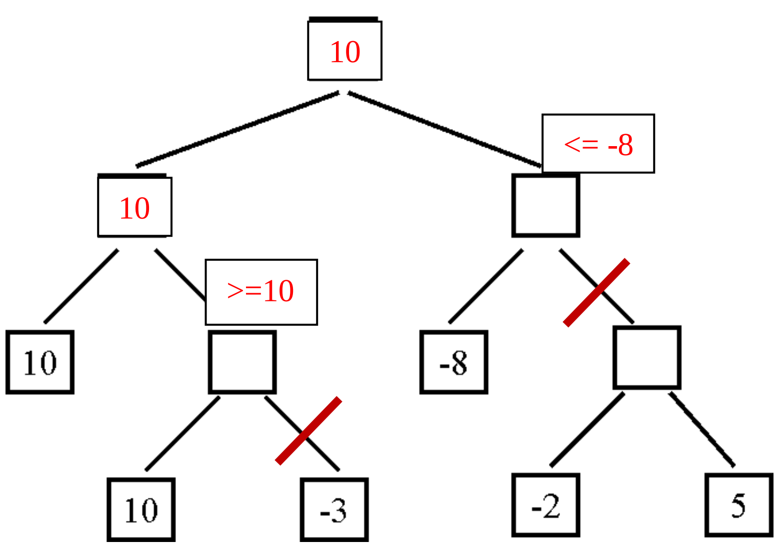
Question 1 (Games) - 30 points

a. (10 points) Trace minimax (i.e., draw the search tree and show the utility value for each terminal and non-terminal node), starting at the following board state, and assuming that X makes the next move. Utility values are +1 if X wins, 0 for a tie, and -1 if O wins.

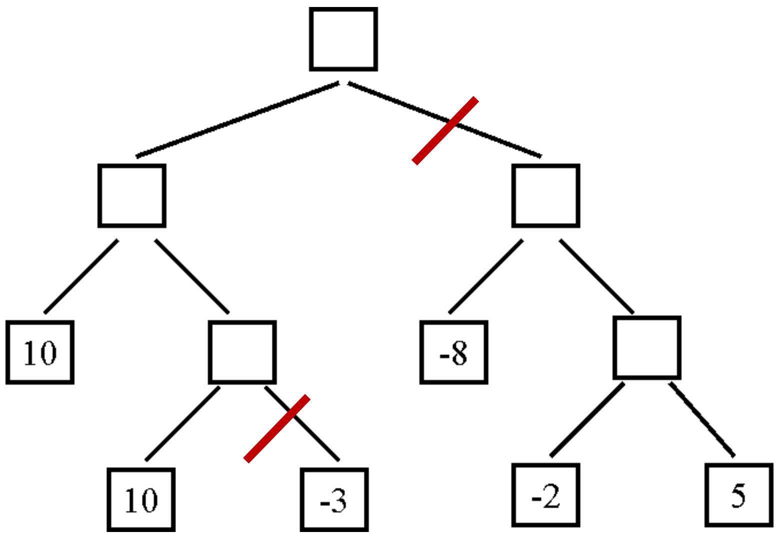
Starting board state:

X	X	O
X	O	O

b. (10 points) In the search tree below, indicate what nodes will be pruned using alpha-beta search, and what the estimated utility values are for the rest of the nodes. Assume that, when given a choice, alpha-beta search expands nodes in a left-to-right order. Assume that the max player plays first.



c. (10 points, harder) The search tree below is the same as the one shown for question 3b. For this question (question 3c only) we are given some additional knowledge about the game: the maximum utility value is 10, i.e., it is not mathematically possible for the MAX player to get an outcome greater than 10. How can this knowledge be used to further improve the efficiency of alpha-beta search? Indicate below the nodes that will be pruned using this improvement.



Question 2 - 30 points

a. (5 points) Consider the following knowledge base:

A AND B
C OR D
(A => (C OR D)) AND (NOT (A => C))

How many rows are there in the truth table for this knowledge base? How did you determine this number?

There are four symbols, A, B, C, D, so there are 2^4=16 rows in the truth table.

b. (5 points) Which sentences, if any, do you obtain by applying the resolution inference rule to the following pair of sentences? **Do not do any simplifications to either the input or the output sentences, just blindly apply the resolution rule.**

A OR (NOT B) OR C OR (NOT D)
(NOT B) OR (NOT C) OR D OR H

1. A OR (NOT B) OR (NOT D) OR (NOT B) OR D OR H
2. A OR (NOT B) OR C OR (NOT B) OR (NOT C) OR H

c. (5 points) Which sentences, if any, do you obtain by applying the resolution inference rule to the following pair of sentences? **Do not do any simplifications to either the input or the output sentences, just blindly apply the resolution rule.**

A OR (NOT B) OR C OR (NOT D)
(NOT B) OR C OR (NOT G) OR H

The resolution rule is not applicable here.

d. (5 points) Put the following knowledge base in conjunctive normal form:

A => (B OR NOT C)
C OR (A AND (NOT B))

(NOT A) OR B OR (NOT C)
C OR A
C OR (NOT B)

e. (5 points) John and Mary sign the following binding contract in front of their parents:
1. On Sunday, John will mow the lawn or buy groceries.
2. On Sunday, Mary will mow the lawn or wash the car.

This is an all-inclusive list of what actually happens on Sunday:

1. Mary mows the lawn on Sunday.
2. Mary washes the car on Sunday.
3. Mary buys groceries on Sunday.

How can the above statements be represented using propositional logic? First, define literals and specify what English phrase each literal corresponds to. Second, represent the knowledge base (i.e., what happens on Sunday) using those literals. Third, represent the contract as a single logical statement, using those literals. Four, determine (in any way you like) whether, according to the rules of propositional logic, the contract was violated or not.

We define the following literals:
JM: John mows the lawn on Sunday
JB: John buys groceries on Sunday
MB: Mary buys groceries on Sunday
MM: Mary mows the lawn on Sunday
MW: Mary washes the car on Sunday

This is what happens on Sunday:
MM and MW and MB and (not JM) and (not JB)

Note: (not JM) and (not JB) are included because the question explicitly says that the given list of what happens on Sunday is all-inclusive (so, whatever is not specified there, did not happen).

The contract is represented as:
(JM or JB) and (MM or MW)

The contract was violated, since (JM or JB) was false.

f. (5 points, harder) Suppose that a knowledge base contains only symbols A, B, and C. When does such a knowledge base entail the statement D (i.e., the statement consisting of a single symbol that does not appear in the knowledge base)? Always, sometimes, or never? If sometimes, then identify precisely the conditions that determine whether this knowledge base entails the statement D.

There is only one case where this knowledge base entails D: the case where the knowledge base is always false. A knowledge base that is always false entails everything.

Question 3 – 15 points

Conjunctive Normal Forms

a. (7 points) Put the following propositional-logic knowledge base in conjunctive normal form:

((NOT A) AND B) => (C OR A)
A AND B AND (C OR D)

A OR (NOT B) OR C OR A
A
B
C OR D

b. (8 points) Suppose that some knowledge base contains various propositional-logic sentences that utilize symbols A, B, C, D, E (connected with various connectives). There are only two cases when the knowledge base is false:
- First case: when A is true, B is true, C is false, D is false, E is false.
- Second case: when A is true, B is false, C is false, D is false, E is true.
In all other cases, the knowledge base is true. Write a conjunctive normal form for the knowledge base. (Hint: there is a much simpler and quicker way to get the right answer, compared to considering all 32 entries in the truth table).

(NOT A) OR (NOT B) OR C OR D OR E
(NOT A) OR B OR C OR D OR (NOT E)

Question 4 - 15 points

Logical Equivalence

Determine if the following pairs of sentences are logically equivalent, meaning that one is true if and only iff the other is true. You do not have to justify your answer.

a. (5 points) Propositional logic.

```
A or B or not(B) or C
A or B or (C <=> C)
```

Logically equivalent, they are both always true

b. (5 points) First-order logic, x and y are variables, f is a predicate.

```
for-every x, for-every y: f(x, y)
for-every x, for-every y: f(y, x)
```

Logically equivalent, "for-every x, for-every y" is the same as for-every y, for-every x, so:

for-every x, for-every y: f(x, y) is the same as
for-every y, for-every x: f(x, y)

By consistently replacing x with y and y with x in "for-every y, for-every x: f(x, y)", we obtain "for-every x, for-every y: f(y, x)"

c. (5 points) Propositional logic.

```
(A and B) => (E and G)
not(A) or not(B) or (E and G)
```

logically equivalent, the second statement is obtained from the first one by applying the rule that "X => Y" is the same as "(not X) or Y"