Quiz Section 1: Propositional Logic Translation

Task 1 – Lost in Logical Translation

For each of the following sentences, define atomic propositions and write a symbolic representation of the sentence.

a) If I am lifting weights this afternoon, then I do a warm-up exercise.

b) If I am cold and going to bed or I am two-years old, then I carry a blanket.

c) Being a U.S. citizen and over 18 is sufficient to be eligible to vote.

d) Unless I am trying to type something, my cat is either eating or sleeping.

Task 2 – Contrapositive Vibes Only

For each of the following sentences:

1. Define a set of atomic propositions and then use them to translate the sentence into logic.
2. Write the **contrapositive** of that logical statement.
3. Rewrite the statement so that all "$\neg$" symbols are next to atomic propositions.
4. Translate the resulting statement back into English.

For Step 3, you can rewrite an expression of the form "$\neg(P \wedge Q)$" as "$\neg P \vee \neg Q$" because the two expressions always have the same truth value. (They are "equivalent".) Similarly, you can rewrite an expression of the form "$\neg(P \vee Q)$" as "$\neg P \wedge \neg Q$". The fact that these pairs of expressions are equivalent are known as De Morgan's Laws. We will see more examples in Topic 2.

a) If I am not bored, then we have class or the sun is out.

b) I have finished reading the book if it has been out for a week and I don't have homework.

Task 3 – The Great Fruit Conspiracy

There are three boxes, one contains only apples, one contains only oranges, and one contains both apples and oranges. The boxes have been incorrectly labeled such that no label identifies the actual contents of its box. Opening just one box, and without looking in the box, you take out one piece of fruit. By looking at the fruit, how can you immediately label all of the boxes correctly?

a) Fill in the table showing all the logical possibilities for what could be in each of the boxes. Then indicate which possibilities are consistent with the description in the problem.

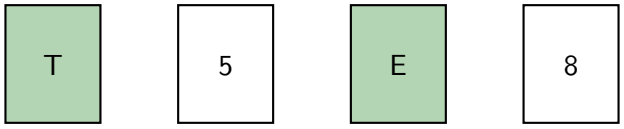
Label: Apples	Label: Oranges	Label: Both	All Labels Incorrect

b) If you take a fruit from the box labelled "Apples", will you always know what is in the other boxes? Why or why not?

c) How do you solve the problem?

Task 4 – Don't get Carded

In this game, four *two-sided* (one green, one white) cards will be placed on the table. On the green side of each card is a letter, and on the white side is a number. For example, you could see the following:



While either color of card can be placed first, the four cards must *alternate* between green and white. In addition, they must follow these rules:

1. If this card is even, then the next card must be a consonant.
2. If this card is a consonant, then the next card must be odd.
3. If next card is consonant, then this card must be even.

a) If the first card is even, what cards possibly can appear last?

b) If the first card is odd, what cards possibly can appear last?

Task 5 – Tabled until Future Notice

Write a truth table for each of the following:

a) $(p \oplus q) \vee (\neg r \wedge p)$

b) $(r \vee q) \rightarrow (r \oplus q)$

c) Write the canonical CNF and DNF expressions in propositional logic for the table from part (b).

d) Consider the function B that takes boolean parameters r and q. Write the canonical DNF expression for part b in Java. The code should look as follows:

```
public static boolean B(boolean r, boolean q) {
    if (______)
        return true;
    if (______)
        return true;
    if (______)
        return true;
}
```

Task 6 – Mission: SATpossible

> You are trying to crack a code consisting of three digits, each between 1 and 5. You started out by randomly trying the codes 533, 353, 131, 422, and 213. In each case, the screen says that the code was wrong but at least one digit was correct (and in the right place).
>
> But now, the screen says that you have only 1 more chance. If the code you enter is wrong, the machine will self-destruct!

You could definitely brute-force this problem by listing all possible codes and checking whether each one satisfies all the constraints (i.e., that it shares at least one digit in the same position with each guess). Since the code has three digits, each between 1 and 5, there are $5^3 = 125$ possible combinations to check. That's manageable in this case, but imagine a problem with a much larger search space—brute-forcing would become extremely inefficient. So instead, we'll explore a more powerful and scalable method: solving it using a SAT solver!

Recall from lecture that the SAT solver takes a logical expression in CNF form as input and returns a way to assign truth values to the variables so that the entire expression evaluates to true. Since CNF is "**AND**s of **OR**s", our goal is to encode each of the constraints as an **OR** statement, and then combine all of them together using **AND** to form the final input for the SAT solver. The SAT problem will have 15 boolean variables. $v_1, \ldots, v_5$ will indicate whether the first digit is 1–5, respectively. $v_6, \ldots, v_{10}$ will indicate whether the second digit is 1–5, respectively. $v_{11}, \ldots, v_{15}$ will indicate whether the third digit is 1–5, respectively. All of our constraints will be expressed using these 15 variables.

a) We will start by creating constraints to ensure that *exactly one* of $v_1, \ldots, v_5$ is true. (We can't have, for example, both v_1 and v_2 to be true since the first digit can't be both 1 and 2.)

The simplest way to do that is to have one formula that requires at least one of these five variables to be true and then additional formulas for each unique pair of variables requiring that at least one of them is false (i.e., they are not **both true**). If all of those formulas are satisfied, then exactly one variable must be true.

Hint: For five variables, there are $4 + 3 + 2 + 1 = 10$ unique pairs. For each pair, how can we express the idea that *they are not both true* using a disjunction? You may find *De Morgan's Laws* from Task 2 helpful :)

Write out the 11 formulas just described that ensure that exactly one of $v_1, \ldots, v_5$ is true.

We can use **AND** to combine the formulas together to form a CNF. By repeating the formula above for the other two digits (i.e., $v_6, \ldots, v_{10}$ and $v_{11}, \ldots, v_{15}$), we can ensure that for each digit of the code, exactly one of the values 1 through 5 is true. The full CNF formula (so far) will then contain a total of 33 conjuncts: 11 conjuncts for each of the 3 digit positions.

We also need to ensure that, for each of the 5 guesses mentioned above, at least one digit is correct (i.e., at least one corresponding variable is set to true). We will start with the first guess, 533.

b) Write out a single formula that ensures that 533 has at least one digit correct.

By repeating the formula above for the other four guesses (i.e., 353, 131, 422, 213), we can ensure that every guess has at least one correct digit. This requires a total of 5 conjuncts.

c) In summary, repeating the construction from part (a) for the three sets of 5 variables gives 33 conjuncts. Repeating the construction from part (b) for each of the 5 guesses tested so far adds 5 more conjuncts. So we have 38 conjuncts in total.

Run the SAT solver on your CNF input with 15 variables and 38 conjuncts:

https://homes.cs.washington.edu/~kevinz/sat-solver/

It has instructions for how to write your CNF formula in the DIMACs format required by the tool.

Show the input you gave it and the output it produced. Then, explain what the output means (i.e., what is the correct code).