PROBLEM 1  English to logic

Rewrite each of the following English sentences as an expression over propositions. Include both a mapping from symbols to propositions and the final expression (see the example). If there are ambiguities, explain where they arise, and give two non-equivalent interpretations.

1. (example) If I forget my keys I can’t get into the house unless my roommate is home.
   \[ K: \text{I remember my keys} \]
   \[ H: \text{I can enter my house} \]
   \[ R: \text{My roommate is home} \]
   \[ (\neg K \land \neg R) \rightarrow \neg H \]

2. I prefer oranges to apples, although apples are less messy to eat
   \[ P: \text{I prefer oranges to apples} \]
   \[ M: \text{apples are less messy than oranges} \]
   \[ P \land M \]

3. If you can prove \( P \neq NP \) (or \( P = NP \), though I hope you don’t), you’ll become famous and I’ll give you an A in this class
   \[ E: \text{You can prove } P = NP \]
   \[ N: \text{You can prove } P \neq NP \]
   \[ F: \text{You’ll be famous} \]
   \[ A: \text{I’ll give you an A} \]
   \[ H: \text{I hope } E \]
   \[ ((E \lor N) \rightarrow (F \land A)) \land \neg H \]
   \[ \text{note: the parenthetical is a separate claim, implicitly anded with others} \]

4. Python programmers must be lazy because Python programs are so much shorter than the equivalent Java or C++ programs
   \[ L: \text{Python programmers are lazy} \]
   \[ J: \text{Python programs are shorter than Java programs} \]
   \[ C: \text{Python programs are shorter than C++ programs} \]
   \[ (J \land C) \rightarrow L \]
   \[ \text{note: the “or” linguistically means “and” logically in this case} \]
**Problem 2 If Statements**

Write an expression for when the following function returns the given return values. Use the variables `a` and `b` as your propositions.

```python
def f(a, b):
    if a:
        return "left"
    elif b:
        return "right"
    else:
        return "up"
```

Returns "right" when \( b \land \neg a \)

Returns "up" when \( \neg b \land \neg a \)

**Problem 3 Truth Tables**

Fill in the following truth tables

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>((A \lor C) \leftrightarrow (B \land C))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 1 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1 0 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0 1 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1 1 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0 1 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 0 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>(((A \oplus B) \lor (A \oplus C)) \lor (B \oplus C))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1 1 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1 1 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1 1 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0 1 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 0 0</td>
</tr>
</tbody>
</table>

In each of the blanks below, put 1\(^{st}\) if the first truth table above is the given idea; 2\(^{nd}\) if the second truth table is; leave it blank if neither is.

___ at least one of \( A, B, \) and \( C \) is 1

___ at least one of \( A, B, \) and \( C \) is 0

___ \( A, B, \) and \( C \) are all the same

2\(^{nd}\) \( A, B, \) and \( C \) are not all the same

1\(^{st}\) either \( A \) and \( C \) are both false or \( B \) and \( C \) are both true, but not both

1\(^{st}\) either \( A \) and \( C \) are both false or \( B \) and \( C \) are both true, or both