Let $L \in L_1 \bigoplus L_2$. We want to prove that $L \in \mathrm{NP} \cap \mathrm{coNP}$. First of all, if L_1 and L_2 are in NP and coNP, then that means there exist non-deterministic Turing machines that decide L_1, $\overline{L_1}$, L_2, and $\overline{L_2}$ in polynomial time. We name these machines $1\mathrm{M}$, $\overline{1\mathrm{M}}$, $2\mathrm{M}$, and $\overline{2\mathrm{M}}$, respectively. Such machines exist because of the aforementioned definition of NP, and that the definition of a language being coNP is if its complement is in NP (i.e. if its complement can be decided by such a machine). This gives rise to our following algorithm:

To show $L \in \mathrm{NP}$:
If $\left((L \in L_1) \wedge (L \notin L_2)\right)$, i.e. $(L \in L_1) \wedge (L \in \overline{L_2})$, we output yes if and only if $1\mathrm{M}$ accepts and $\overline{2\mathrm{M}}$ accepts. Then if $\left((L \in L_2) \wedge (L \notin L_1)\right)$, i.e. $(L \in L_2) \wedge (L \in \overline{L_1})$, we output yes if and only if $\overline{\mathrm{AM}}$ accepts and $\mathrm{BM}(\mathrm{L})$ accepts.

Similarly, to show that $L \in \mathrm{coNP}$:
If $\left((L \in L_1) \wedge (L \notin L_2)\right)$, we output yet if and only if $\overline{1\mathrm{M}}$ accepts and $2\mathrm{M}$ accepts. Then if $\left((L \in L_2) \wedge (L \notin L_1)\right)$, we output yes if and only if $1\mathrm{M}$ accepts and $\overline{2\mathrm{M}}(\mathrm{L})$ accepts.