Exercise 1: Argue that if $\Delta_k^{\mathbf{P}} = \Sigma_k^{\mathbf{P}}$, then $\Delta_k^{\mathbf{P}} = \mathbf{PH}$.

Solution: My plan is to prove that if $\Delta_k^{\mathbf{P}} = \Sigma_k^{\mathbf{P}}$, then $\Delta_k^{\mathbf{P}} = \Pi_k^{\mathbf{P}}$. This implies that $\Sigma_k^{\mathbf{P}} = \Pi_k^{\mathbf{P}}$, and we have seen in class that this implies $\Sigma_k^{\mathbf{P}} = \Pi_k^{\mathbf{P}} = \mathbf{PH}$.

For this, recall the recursive definition of these classes:

$$\Delta_k^{\mathbf{P}} = \mathbf{P}^{\Sigma_{k-1}^{\mathbf{P}}}.$$
$$\Sigma_k^{\mathbf{P}} = \mathbf{NP}^{\Sigma_{k-1}^{\mathbf{P}}}.$$
$$\Pi_k^{\mathbf{P}} = \mathbf{co\text{-}NP}^{\Sigma_{k-1}^{\mathbf{P}}}.$$

The inclusions $\Delta_k^{\mathbf{P}} \subseteq \Sigma_k^{\mathbf{P}}$ and $\Delta_k^{\mathbf{P}} \subseteq \Pi_k^{\mathbf{P}}$ clear from the definitions and have been discussed in class. Therefore, it is enough to prove

$$(1) \qquad\qquad \Sigma_k^{\mathbf{P}} \subseteq \Delta_k^{\mathbf{P}} \implies \Pi_k^{\mathbf{P}} \subseteq \Delta_k^{\mathbf{P}}.$$

Suppose that $\Sigma_k^{\mathbf{P}} \subseteq \Delta_k^{\mathbf{P}}$ and let $X \in \Pi_k^{\mathbf{P}}$ be a language. Then, $\overline{X} \in \Sigma_k^{\mathbf{P}} \subseteq \Delta_k^{\mathbf{P}}$. However, $\Delta_k^{\mathbf{P}}$ is closed under compliment, because its languages can be decided deterministically in polynomial time with a $\Sigma_{k-1}^{\mathbf{P}}$ oracle (in particular, the output bit can just be flipped). More formally, let M be a DTM that decides $\overline{X}$ with access to a $\Sigma_{k-1}^{\mathbf{P}}$ oracle. Then, a DTM M' that decides X with access to the same oracle can be constructed: Run M on the input and output the opposite of the answer given by M. This implies that $X \in \Delta_k^{\mathbf{P}}$, proving (1).

Exercise 2: Let $A, b \subseteq \{0,1\}*$ be two languages. Show that if $A \leq_m^l B$ and $B \in \mathbf{NL}$, then $A \in \mathbf{NL}$.

Solution: Recall that $A \leq_m^l B$ means that there is a log-space computable function f (say, by a deterministic Turing machine D) such that

$$(2) \qquad\qquad x \in A \iff f(x) \in B.$$

On the other hand, $B \in \mathbf{NL}$ means that there is a log-space non-deterministic Turing machine M such that M accepts x if and only if $x \in B$. Naïvely, a nondeterministic turing machine N can be constructed that computes $f(x)$ by running D and runs M on the output. However, this machine would not necessarily run in log-space, because it would need to write the output of f on its work tape in order for M to read it. Instead, use a trick introduced in class: I construct the kth bit of $f(x)$ whenever it is needed by M. To achieve this, in addition to the input tape, the output and work tapes of M, and the work tapes of D, I need:

- A work tape K of which only one bit is used, which corresponds to the kth bit of $f(x)$. This serves as the output tape of D and the input tape of M. The head associated with this tape never moves.

- A work tape MC to store the position of the read head of M, and increment it or decrement it whenever the program of M moves the read head.

- A work tape DC to store the position of the write head of D, and increment it or decrement it whenever the program of D moves the write head.

More precisely, the machine works as follows:

Algorithm 1 Nondeterministic Turing Machine N that decides A

Require: x
1: $MC \leftarrow 0$
2: **for** step S of a run of M **do**
3:     **if** S reads from the input **then**
4:        $DC \leftarrow 0$
5:        **for** step T of a run of D on the input x from its initial state **do**
6:           **if** T writes bit b to the output tape and $DC = MC$ **then**
7:              Write b in K
8:           **end if**
9:           Perform step T, without moving N's write head or writing to its output tape
10:          **if** T moves the write head to the left **then**
11:             $DC \leftarrow DC - 1$
12:          **else if** T moves the write head to the right **then**
13:             $DC \leftarrow DC + 1$
14:          **end if**
15:        **end for**
16:     **end if**
17:     Perform step S, reading from K instead of the input tape, and never moving N's read head
18:     **if** S moves the read head to the left **then**
19:        $MC \leftarrow MC - 1$
20:     **else if** S moves the read head to the right **then**
21:        $MC \leftarrow MC + 1$
22:     **end if**
23: **end for**

Note that the machine N is a (non-deterministic, because the steps of the machine M used are non-deterministic) Turing machine that completely emulates the machine M on the input $f(x)$. Therefore, it accepts x if and only if M accepts $f(x)$, that is, if and only if $x \in A$. Furthermore, it only uses as much space as M and D together, plus the space used in the tapes MC, DC and K. However, K is only one bit, and MC and DC can count only up to $|f(x)|$, which is polynomial in $|x|$, because D is a log-space machine. Therefore, the number of bits used by N is logarithmic in $|x|$. This concludes the proof that $A \in \mathbf{NL}$.