(1) (a) If \( \sum_{i \in I} x_i = t \) for some \( I \subseteq \mathbb{N} \), then

\[
\sum_{j \notin I} x_j = \sum_{i=1}^{n} x_i - \sum_{i \in I} x_i = y - t
\]

So

\[
\sum_{i \in I} x_i = t
\]

\[
y - t = \sum_{j \notin I} x_j
\]

so adding the two sides we have

\[\langle x_1, \ldots, x_n, y-t, t \rangle \in \text{PARTITION}\]

(b) There are 2 problems:
(i) \( y-t \) can be negative: e.g., \( N=1 \)
\((x_1, t) = (1, 2)\) so \(y = \sum_{i=1}^{n} i = 1\), so \(y - t = 1 - 2 < 0\).

(ii) There is another way to sum \(x_1, \ldots, x_n, y - t, t\) to get equality whether or not \(\langle x_1, \ldots, x_n, t \rangle \in \text{SUBSET-SUM}\); namely: \(N=1, x_1=3, t=2\), so \(\langle x_1, t \rangle \notin \text{SUBSET-SUM}\), but \(y = 3 - 2 = 1\) and \(x_1 = 3 = 2 + 1 = t + (y - t)\).

(C) The trick is to choose \(B\) so that problems (i) and (ii) can’t happen; so we need

(i) \(y - t + B \geq 1\), so \(B \geq 1 + t - y\), and

(ii) \(y - t + B\) and \(t + B\) must be on opposite sides of the partition, so that
\( \langle x_1, \ldots, x_n, y-t+B, t+B \rangle \in \text{PARTITION} \)

\[ \Rightarrow \quad y-t+B, t+B \text{ are on opposite sides,} \]

i.e. we can't have

\[ \left\{ \begin{array}{l}
(y-t+B) + (t+B) + \sum_{i \in I} x_i = \sum_{j \notin I} x_j \\
\end{array} \right. \]

But for any \( I \subset [N], \)

\[ \sum_{i \in I} x_i \geq 0, \quad \sum_{j \notin I} x_j \leq \sum_{j=1}^{n} x_j = y \]

So implies

\[ (y-t+B) + (t+B) = \sum_{j \notin I} x_j - \sum_{i \in I} x_i \leq y-o = y \]

So

\[ y+2B \leq y \]

which can't happen if \( B \geq 1. \)

So \( \langle x_1, \ldots, x_n, t \rangle \in \text{SUBSET} \)
\[ \langle x_1, \ldots, x_N, y - t + B, t + B \rangle \in \text{PARTITION} \]

if

\[ B \geq \max \left(1 + t - y, 1\right). \]

So it suffices to take

\[ B = \max \left(1 + t - y, 1\right). \]

Then \( B \) is at most \( \max \left(1, t\right) \), so \( y - t + B, t + B \) have their size at most \( y = \sum_{i=1}^{N} x_i \) plus \( t \) plus \( B \). Hence

\[ \left| \langle x_1, \ldots, x_N, y - t + B, t + B \rangle \right| \]

\[ \leq \text{poly} \left( \left| \langle x_1, \ldots, x_N, B \rangle \right| \right). \]

(d) \text{PARTITION } \in \text{NP} \ by \text{ non-deterministically}\]

guessing, on input \( \langle x_1, \ldots, x_N \rangle \), whether \( i \in [N] \)

is put into \( \mathcal{C} \subseteq [N] \) or not, and then

comparing \( \sum_{i \in \mathcal{C}} x_i \) to \( \sum_{j \in N \setminus \mathcal{C}} x_j \).
By (c),

\[ \text{SUBSET-SUM} \leq_p \text{PARTITION}, \]

so \text{PARTITION} is \text{NP-complete}. 
(2) 3COLOR is NP-complete:

In class we explained that $3\text{COLOR} \in \text{NP}$
(by non-deterministically guessing a 3 colouring).

Given a 3CNF in variables $x_1, \ldots, x_N$,
for each variable introduce vertices and edges

Hence if $v_1$ has the colour red (say the
colours are $R = \text{red}$, $G = \text{green}$, $B = \text{blue}$) then each variable must be coloured $G, B$ or $B, G$:

Now we view the top $x_i$ vertex as representing $T$ (true), the bottom as $F$ (false).

For each clause $C_1, \ldots, C_m$ of the 3CNF, we add the following: for a clause $x_i \lor x_j \lor x_k$ we add:

this can be blue iff one of the top $x_i$ or top $x_j$ vertices is blue.

Since
implies one is blue, one red

but, similarly

implies this must be blue,

and allows for can be blue
To enforce that is blue or green, we can add an edge to "red".

\[ \text{enforce green or blue} \]

\[ v_1 \quad \text{or} \quad v_2 \]

2. We now add a similar "OR" gadget between "\( x_i \) or \( x_j \)" and "\( x_k \)".
now we add an edge to insist

hence the rightmost vertex must be blue, i.e. the colour at $v_2$ on the left. We do the same for every other clause, but connecting to the bottom vertex of $X_i$ if $\neg X_i$ appears, and similarly for $X_j, \neg X_j$ and $X_k, \neg X_k$.

Hence we can satisfy the 3CNF iff...
each clause vertex is colourable (with the color at $v_2$, here shown in blue). Hence the 3CNF in question is satisfiable this graph can be 3-coloured.

(3) \textsc{4color} \in \textsc{NP} by non-deterministically guessing a 4-colouring of the graph (i.e. for each vertex connected to at least one edge).

To reduce \textsc{3color} to \textsc{4color}, given a graph $G$, add a new vertex, $v_0$, to $G$, connected to every vertex of $G$ (connected to some edge).

Example

$$G \quad \rightarrow \quad G + v_0 + \text{a bunch of edges}$$
The new graph, $G'$, can be described by having one more vertex, and at most 2 new edges for each edge of $G$. Hence $\langle G' \rangle$ can be generated from $G$ in poly-time.

Any 4-colouring of $G'$ gives a 3-colouring of $G$ with the 3 colours different than the colour of $v_0$; conversely, any 3-colouring of $G$ gives a 4-colouring of $G'$ by colouring $v_0$ with the 4th colour. Hence

$$\langle G \rangle \in 3\text{COLOR} \quad \iff \quad \langle G' \rangle \in 4\text{COLOR}.$$ 

So the map $G \mapsto G'$ gives a reduction $3\text{COLOR} \leq 4\text{COLOR}$.

Hence $4\text{COLOR}$ is NP-complete.
(4)(a) If for some \( i \neq j \), \( \chi_i = \chi_j = T \), then either

1. \( i, j \) are both in \( \{1, \ldots, m\} \)
2. \( i, j \) are both in \( \{m+1, \ldots, n\} \)
3. one each of \( i, j \) are in one of each of \( \{1, \ldots, m\} \) and \( \{m+1, \ldots, n\} \).

Conversely, if \( i \neq j \) are both in \( \{1, \ldots, m\} \) with \( i \neq j \), then \( \text{Th}_{2,n}(\chi_1, \ldots, \chi_n) = T \), and similarly if they are both in \( \{m+1, \ldots, n\} \) or if one is in \( \{1, \ldots, m\} \), the other in \( \{m+1, \ldots, n\} \). Hence the equation holds.

(b) Taking \( m = n/2 \), we have \( \text{Th}_{2,n} \) is written as the \( \wedge \) (AND) of two \( \text{Th}_{2,n/2} \) formulas, and of \( (\chi_1 \lor \ldots \lor \chi_m) \land (\chi_{m+1} \lor \ldots \lor \chi_n) \), the latter of size \( n \)
Hence \( L_n \leq 2 \cdot L_{n/2} + n \).

(c) By induction on \( k \):

for \( k = 1 \), \( n = 2^k = 2 \), \( n \log_2 n = 2 - 1 \), and

\[ T_{T_2}(x_1, x_2) = x_1 \land x_2, \text{ which is of length } 2. \]

Hence this holds for \( k = 1 \).

Assuming it holds for some \( k \), then

\[ L_{2^{k+1}} \leq 2 \cdot L_{2^k} + 2^{k+1} \]

\[ = 2 \cdot 2^k - 1 + 2^{k+1} \]

\[ = 2^{k+1} (k+1) \]

so it holds for \( k+1 \). Hence, by induction, it holds for all \( k \).

(d) \((a, b, c)\) is a slight variant, which is of the same size but is the OR of more clauses. For example,
\((c_1,b,c)\) gives, after rearranging \(x_2, x_3\):

\[
\text{Th}_{2, \delta}(x_1, x_3, x_2, x_4) = \text{Th}_2(x_1, x_3) \lor \text{Th}_2(x_2, x_4) \lor (x_1 \lor x_3) \land (x_2 \lor x_4)
\]

\[
= (x_1 \land x_3) \lor (x_2 \land x_4) \lor (x_1 \lor x_3) \land (x_2 \lor x_4)
\]

But in class we wrote

\[
= \left( (x_1 \lor x_4) \land (x_2 \lor x_3) \right) \land \text{same}
\]

So the \(c_2\) expands to 8 terms

\[
x_1 x_2, x_1 x_3, x_4 x_2, x_4 x_3
\]

Whereas \((a,b,c)\) writes only

\[
x_1 x_3, x_2 x_4
\]

since the other two, \(x_1 x_2, x_4 x_3\), already appear when expanding \(c_1\).
So the formulas are of the same size, but the one in \((a, b, c)\) is the OR of more clauses: they each have a largest clause of size \(N = 2^k\) literals, but the others in \((a, b, c)\) are of size \(2^{kn}\), or \(2^{k-k}\), etc.