1. Prove the following equation by induction over natural numbers.
\[ \forall n \geq 0, (1 \times 2) + (2 \times 3) + \ldots + (n \times (n + 1)) = \frac{n(n+1)(n+2)}{3}. \]

2. Recall some basic definitions about sets and relations.

- A binary relation \( R \) over a set \( S \) is a subset of the cartesian product \( S \times S \). If \( (x, y) \in R \), we say \( x \) and \( y \) are related by \( R \), often written \( xRy \).
- Such a relation is reflexive if \( xRx \) for all \( x \in S \).
- It is transitive if \( xRy \) and \( yRz \) implies \( xRz \) for all \( x, y, z \in S \).
- It is antisymmetric if \( xRy \) and \( yRx \) implies \( x = y \) for all \( x, y \in S \).
- A binary relation is a partial order if it is reflexive, transitive, and antisymmetric.
- The power set \( \mathcal{P}(S) \) of a set \( S \) is the set of all subsets of \( S \).

Now show that for any set \( S \), the set inclusion relation \( \subseteq \) forms a partial order over \( \mathcal{P}(S) \).

3. Consider the following abstract syntax grammar for boolean expressions.

\[ \text{exp ::= True | False | And(exp, exp) | Or(exp, exp) | Not(exp)} \]

Show how to represent these expressions using an algebraic data type in OCaml or Haskell, or a set of case classes in Scala, or an appropriate mechanism in some other language of your choice. Then write a function that evaluates an arbitrary expression of this type to a boolean value using the “obvious” meaning of expression constructors.