Optimising Compilers

Consider the programming language with terms $e$ having abstract syntax:

$$e ::= x | c | \lambda x.e | e_1 e_2 | \text{let } x = e_1 \text{ in } e_2$$

where $x$ ranges over a set of identifiers and $c$ over a set of integer constants. For the rest of the question, your answers can be illustrated by reference to the program $p$:

$$\lambda z.\text{let id} = \lambda x.x \text{ in } id \text{ id } 7$$

State how to label terms in $p$ uniquely so that a subterm occurring repeatedly in a term has different labels. [4 marks]

Show how such terms may be seen as a family of flowgraphs, one for each $\lambda$ (you may find it useful to consider the above labelling as providing a unique function name for anonymous $\lambda$-abstractions). [4 marks]

Define the call graph of such a family of flowgraphs, stating clearly how indirect calls are treated. [4 marks]

Describe how to associate a flow-variable with each labelled node of a term such as $p$ and to derive equations which can improve the above treatment of indirect calls to get a better approximation of the edges in the call graph. [8 marks]

[Hint: you may find it useful to recall the shorthand of $\gamma \rightarrow \delta \supseteq \beta$ as representing the compound constraint that

whenever $(\lambda x^j.e^k)^i \in \beta$ we have $\alpha_j \supseteq \gamma \land \delta \supseteq \alpha_k$

where $\alpha_r$ is the flow variable associated with the node labelled $r$.]