Foundations of Functional Programming

(a) Define the translation of the call-by-name $\lambda$-calculus into continuation passing style. [9 marks]

(b) How does the translation differ for the call-by-value $\lambda$-calculus? [2 marks]

(c) Now consider extending the call-by-name $\lambda$-calculus with exceptions:

\[
M ::= \text{try } M \text{ catch } M \\
| \text{raise} \\
| \lambda x. M \\
| MM \\
| x
\]

where it reduces in the following way:

\[
\begin{align*}
\text{try raise catch } M & \rightarrow M \\
\text{try } \lambda x. M_1 \text{ catch } M_2 & \rightarrow \lambda x. M_1 \\
\text{raise } M & \rightarrow \text{raise}
\end{align*}
\]

Show how to translate this language into pure $\lambda$-calculus using continuations.

[Hint: Use two continuations: one for the exceptional case, and one for the normal case.] [9 marks]