4 Compiler Construction (tgg22)

This question explores how exceptions might be added to SLANG and the JARGON virtual machine. We will raise an exception with

```
raise e
```

where `e` is an expression. We will "trap" an exception with the following expression.

```
try e with f end
```

If `e` evaluates to a value `v`, then `v` is the result of the `try`-expression. Otherwise, the evaluation of `e` raises an exception `E` and the `try`-expression continues by evaluating the function application `f(E)`. To simplify things we will assume that each f is an identifier. Uncaught exceptions at the top-level will result in a runtime error.

(*a*) Do we need to define a fixed type for exceptions? Justify your answer. [3 marks]

(*b*) What typing rule or rules would you implement for the expression `raise e`? Justify your answer. [3 marks]

(*c*) A compiler may rewrite expressions in order to optimise generated programs. For example, here are two rewrite rules to simplify conditional expressions:

	code	replacement
1	`if true then e1 else e2`	`e1`
2	`if false then e1 else e2`	`e2`

For each of the rules below, argue that it is, or is not, a valid optimisation rule.

	code	replacement
1	`raise (raise e)`	`raise e`
2	`e1 + (raise e2)`	`raise e2`
3	`try (raise e) with f end`	`f(e)`
4	`try e with (fun x -> raise x) end`	`e`

[6 marks]

(*d*) Carefully describe the stack-oriented code you would generate for both the `raise`- and `try`-expressions. [8 marks]