1997 Paper 13 Question 10

Introduction to Functional Programming

Consider the following Standard ML type `expression` intended to represent integer arithmetic expressions built up from named variables using addition and multiplication:

```
datatype expression = Var of string
                    | Sum of expression * expression
                    | Product of expression * expression;
```

[In the CAML dialect of ML `datatype` is written `type`.]

For example, the expression $((a + b)c)(d + e)$ would be represented by:

```
Product(Product(Sum(Var "a", Var "b"), Var "c"),
        Sum(Var "d", Var "e"))
```

Write an ML function `freevars` (using any dialect of ML) which takes an argument, `e`, of type `expression` and returns a value of type `string list` containing all the variables in `e`. This list may contain repeated instances of variables.   [10 marks]

Write a second ML function `eval` (again using any dialect of ML) which takes two arguments. The first argument is of type `expression` and the second is an association list of type `(string * int) list` giving a value for each variable, for example:

```
[("a",1), ("b",0), ("c",2), ("d",4), ("e",1)]
```

[In the CAML dialect of ML semicolons are used as list separators.]

When `eval` is applied to the above examples it returns 10.   [10 marks]

1