PRIFYSGOL CYMRU; UNIVERSITY OF WALES

DEGREE EXAMINATIONS MAY/JUNE 2002

SWANSEA

Computer Science

CS 321 Functional Programming II

Attempt 2 questions out of 3

Time allowed: 2 hours

Students are permitted to use the dictionaries provided by the University

Students are NOT permitted to use calculators

CS_321
FUNCTIONAL PROGRAMMING 2

(Attempt 2 questions out of 3)

Question 1

a. What is the general type of a program which can operate on files, and on standard input and output in Haskell or Gofer? What is the assumed type of the operating system? Give brief details of the sorts of values that can be used as elements of the input and output types

[5 marks]

b. Consider the following attempt at defining a simple program in Haskell to perform input and output.

```
main(Success : Str userInput : Success : fCont :_) =
     [ AppendChan stdout "please type a filename",
       ReadChan stdin,
       AppendChan stdout fileName,
       ReadFile fileName,
       AppenChan stdout
          (case fCont of
            Failure ioerror ->
                          "cannot open "++fileName
            Str fileContents -> fileContents
          )
     ] where (fileName:_) = lines userInput
```

Why would the above program fail?

Rewrite it so that it works

i) Using irrefutable patterns.

ii) Without using pattern matching on the input.

iii) Using continuations.

In each case carefully explain how each approach works and discuss the relative merits of the approaches.

[15 marks]

c. Briefly explain what you understand by the concept of a monad and how this concept can be used in programming interactive programs.

[5 marks]

Question 2

Functional programming languages can be implemented by

a. translating to combinators,

b. using an SECD machine, or
c. using data flow graphs.

For each of the above
i) outline each approach,
ii) indicate whether the method implements a lazy or eager evaluation semantics, or whether it can be adapted for either,
iii) indicate what sort of intermediate code is required, and
iv) discuss whether it is suitable for a parallel processor implementation.

[25 marks]

Question 3

a. Given a type system where types can be simple types, functions or lists define a Haskell/Gofer data type to represent type expressions. Why might it be necessary to extend this to define Type Schemes?

[3 marks]

b. Define Haskell/Gofer data types to represent expressions in a simple language based on the lambda calculus where expressions can be variables, function abstractions, or function applications.

[3 marks]

c. Explain what is meant by

i) Term unification, and
ii) Most general unifier

And describe how type checking can be considered to be an application of term unification. Your answer should refer to the data types defined in **(a)** and **(b)** and show how they are used.

[9 marks]

d. Outline how Type Classes have been defined in Haskell and Gofer to handle ad-hoc polymorphism. Your answer should include examples of how a hierarchy of classes can be defined and an indication of how dictionaries can be used to implement type classes.

[10 marks]