PRIFYSGOL CYMRU; UNIVERSITY OF WALES

DEGREE EXAMINATIONS MAY/JUNE 2002

SWANSEA

Computer Science

CS 221 Functional Programming I

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_221
FUNCTIONAL PROGRAMMING 1

(Attempt 2 questions out of 3)

Question 1

a. Explain the difference between *datatypes* and *type synonyms* as defined in Haskell and Gofer. Give examples of a type synonym, a datatype defined using another type, and a datatype defined without using another datatype, and comment on the advantages and disadvantages of each approach.

[5 marks]

b. Define a general datatype and a reduction function for strict binary trees (*ie* nodes may only have 0 or 2 successors) which can contain values of one type at the leaves and values of a different type at the nodes within the tree. Derive the type for the reduction function.

[6 marks]

c. How could you extend the definitions in **(b)** to include trees where nodes may have 0, 1 or 2 successors?
Define a `sort` function that when given a list of values and an appropriate ordering function would return a binary tree such that at any node in the tree all values in the left subtree would be less than the value at the node, and all values in the right subtree would be greater than or equal to the value at the node.

[8 marks]

d. Given a tree generated by the above `sort` function define functions to
i) output the values of the tree in increasing order using the general reduction function defined in **(c)**.
ii) output the values of the tree in decreasing order without using the general reduction function defined in **(c)**.

[6 marks]

Question 2

a. What is meant by the following terms?

i) Normal form
ii) Leftmost reduction sequence
iii) Lazy evaluation
iv) Fixed point operator
v) Referential Transparency

[5 marks]

b. Consider the following function and explain why its recursive nature might be a problem in a simple-minded approach to functional language implementation. Give an alternative definition removing the recursion.

```
factorial n = if n <= 1 then 1 else (factorial (n-1))*n
```

[4 marks]

c. Reduce to normal form (if possible) the following expressions:

i) (λxyz.xz(yz)) (λxy.x) (λxy.x)
ii) (λx.((λz.zx)(λx.x)))y
iii) (λx.((λy.xy)z))(λx.xy)
iv) (λaf.(f a))(λg.(g g)) (λs.(s s))

[8 marks]

d. Prove that if G=λyf.f(yf) then M is a fixed-point operator if and only if M=GM

Hence show that Z = (λxy.y(xxy))(λxy.y(xxy)) is a fixed-point operator.

[8 marks]

Question 3

a. Define the following functions and derive their types.

i) `map f xs` which applies the function `f` to every element of a list `xs`

ii) `filter p xs` which returns the list containing all those elements `x` in `xs` for which `p x` is true

iii) `palin xs` which returns a pair `(b,a)` where if `xs` is palindromic (reads the same backwards and forwards) and contains more than 1 entry then `b` is True, otherwise `b` is False, and if `b` is True then `a` is the length of `xs` otherwise it is 0.

[9 marks]

b. In the standard prelude for Haskell there is a function `words xs` which given a string containing a number of words separated by spaces returns a list of words, eg

```
words "the words are" = ["the","words","are"]
```

Using this function define a function which will return either

- the length of the longest palindrome in a string containing a number of words separated by spaces, or
- the number of words in the string if there are no palindromic words in it.

[6 marks]

c. Prove that for any function `f`, any total predicate `p` and any finite list `xs`

```
filter p (map f xs) = map f (filter (p.f) xs)
```

[10 marks]