CS-321
FUNCTIONAL PROGRAMMING 2
(Attempt 2 questions out of 3)

Question 1

a. A functional language can be implemented by translating it into an extended version of λ-calculus which includes basic arithmetic and logical operations. Give typical translations of the following pieces of Haskell code into λ-expressions. Function abstractions should contain only bound variables and constant identifiers.

i)
```
sumsquares x y = (square x) + (square y)
                 where square z = z * z
```

ii)
```
sum n = if n == 1 then 1 else n + sum (n-1)
```

[5 marks]

b. Assume that the following constants have the following types:-

```
+:     int -> int -> int
-:     int -> int -> int
FIX:   ((a->b)->(a->b))->(a->b)
COND:  bool -> c -> c -> c
EQ:    d -> d -> bool
1:     int
```

Carefully derive the type of sum by assigning appropriate types to each component of the λ-translation you gave in part **(a)**. Indicate how, and where, type equivalences arise.

[5 marks]

c. Explain how the type checking process you carried out in **(b)** might be considered as an example of **term unification**. Explain the terms **unifier**, **most general unifier**, and **substitution** in the context of term unification and outline how these ideas might be used to implement a type checker in Haskell. Give examples of the functions and types that would need to be defined. Full definitions of functions are not required, but you should give their type and outline their purpose.

[10 marks]

d. Consider the following type definitions

```
member :: Eq a => [a] -> a -> Bool
example :: Ord b => [[b]]
```

Outline how the expression `member example` would be type checked and give the type that would be derived.

[5 marks]

Question 2

a. What is the general form for a class declaration? Make sure you explain the role of each component of the declaration.

[4 marks]

b. How can a type be defined to be an instance of a class? Give an example definition.

[4 marks]

c. Type classes can be implemented using the concept of a dictionary with up to three components. What are these components? For each component give an example of a dictionary where that component is empty.

[6 marks]

d. Describe how a functional programming language can be implemented using **EITHER** a *SECD machine*, **OR** the *data flow* approach.

[8 marks]

e. For the implementation method you have described in the answer to part **(d)** indicate whether it implements a strict language or a lazy language. Discuss how it might be adapted to implement a lazy language if initially it implemented a strict language, or vice versa.

[3 marks]

Question 3

a. Briefly define/explain the following terms

i) leftmost redex

ii) normal order reduction

iii) lazy evaluation

iv) eager evaluation

[4 marks]

b. What is the difference between **free** and **bound** variables? Indicate which instances of variables in the following expression are free and which are bound and why. Also identify all redexes indicating whether they are β or η redexes – (λa.((λb.bb) a)) ((λd.d) a)

[7 marks]

c. Böhm's theorem states that if G=λyf.f(yf) then M is a fixed-point operator if and only if M=GM. Use this to show that Z = (λxy.y(xxy))(λxy.y(xxy)) is a fixed-point operator.

[4 marks]

d. One way of evaluating the λ-expressions is to translate them into combinators. Assume that some expression has been translated into the following combinator code.

```
S ( S (K *) I ) I 2
```

Draw the graph to represent this code and indicate how it would be evaluated. A complete evaluation is not required, but you should outline the steps involved in the evaluation process. What evaluation strategy does the process you describe implement? Would it be possible to adapt it to implement an alternative strategy?

[10 marks]