COMPUTER SCIENCE TRIPOS Part IA

NATURAL SCIENCES TRIPOS Part IA (Paper CS/1)

POLITICS, PSYCHOLOGY, AND SOCIOLOGY TRIPOS Part I (Paper 9)

Monday 6 June 2011 1.30 to 4.30

COMPUTER SCIENCE Paper 1

*Answer **one** question from each of Sections A, B and C, and **two** questions from Section D.*

*Submit the answers in five **separate** bundles, each with its own cover sheet. On each cover sheet, write the numbers of **all** attempted questions, and circle the number of the question attached.*

You may not start to read the questions printed on the subsequent pages of this question paper until instructed that you may do so by the Invigilator

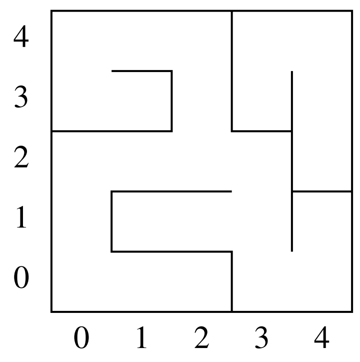
STATIONERY REQUIREMENTS
Script paper
Blue cover sheets
Tags

SPECIAL REQUIREMENTS
Approved calculator permitted

SECTION A

1 Foundations of Computer Science

This question considers 2-dimensional rectangular labyrinths such as the following.



Here horizontal and vertical bars indicate *walls*. One can *step* from a position to an adjacent position if there is no wall obstructing the move. For example, one can move *in one step* from position (1,1) to position (2,1), but not to (1,0). Diagonal steps are not allowed.

(*a*) Carefully explain a convenient way of representing such labyrinths in ML. Then define a function, call it `next`, which takes two arguments, a position `(x,y)` and a labyrinth, and returns a list of all positions that can be reached *in one step* from position `(x,y)`. [Hint: Consider representing labyrinths as functions with a type of the form `int * int ->` *something*.] [8 marks]

(*b*) Use `next` to code, in ML, a space-efficient function that checks whether it is possible to move from a position `(x1,y1)` to another position `(x2,y2)` in a given labyrinth. For full marks, make sure your function always terminates. [6 marks]

(*c*) Explain, not necessarily using ML code, how one can code a function that returns the *shortest path* between two given positions. Here path means a list of all the positions one would visit en route to the destination. The solution must be space efficient and practical even for large labyrinths. Briefly explain why your solution is space efficient. [6 marks]

2 Foundations of Computer Science

(*a*) Write brief notes on exceptions in ML and on the functions and control structures available for programming with them. [6 marks]

Parts (*b*) and (*c*) make use of the following ML exception:

```
exception Olive;
```

(*b*) Code in ML a function called `cannot` which takes two arguments, a function `f` and a value `x`. Define the `cannot` function in such a way that it returns `true` if and only if evaluation of `f(x)` causes exception `Olive`. For all other inputs, it should return `false`. [Hint: evaluation of `f(x)` may cause exceptions other than `Olive`.] [4 marks]

(*c*) Consider the following ML datatype and functions `bun` and `cheese`.

```
datatype 'a tree = Leaf of 'a
                 | Branch of 'a tree * 'a tree;

fun bun (x,Leaf y)         = if x=y then raise Olive else Leaf y
  | bun (x,Branch (t1,t2)) = Branch (bun(x,t1),bun(x,t2))

fun cheese (x,t) = if cannot(bun,(x,t)) then Leaf x else bun(x,t)
```

(*i*) Write down the type of `cheese`. [3 marks]

(*ii*) Write a function that is equivalent to `cheese` but makes no use of exceptions. Briefly explain why your function is equivalent to `cheese`. [7 marks]

SECTION B

3 Discrete Mathematics I

This question is about structured proofs.

(*a*) Write down the introduction and elimination rules for implication and negation. [4 marks]

(*b*) Using the rules from part (*a*), give a structured proof of

$$(P \Rightarrow Q) \Rightarrow ((\neg Q) \Rightarrow (\neg P))$$ [7 marks]

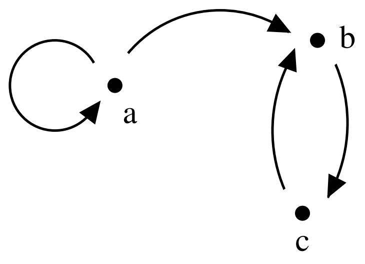
(*c*) Write down the rule for proof by contradiction. [2 marks]

(*d*) Using the rules from parts (*a*) and (*c*), give a structured proof of

$$((\neg Q) \Rightarrow (\neg P)) \Rightarrow (P \Rightarrow Q)$$ [7 marks]

4 Discrete Mathematics I

Consider the following directed graph.



(*a*) Write down a set of ordered pairs that describes the graph. [3 marks]

(*b*) Consider the following four formulae about a relation R.

(*i*) $\forall x.\,(x,x) \in R$

(*ii*) $\forall x.\forall y.\forall z.\Big(\big((x,y) \in R \ \wedge\ (y,z) \in R\big) \ \Rightarrow\ (x,z) \in R\Big)$

(*iii*) $\forall x.\forall y.\Big((x,y) \in R \ \Rightarrow\ \exists z.\big((x,z) \in R \ \wedge\ (z,y) \in R\big)\Big)$

(*iv*) $\forall x.\exists y.\big((x,y) \in R \ \Rightarrow\ \forall z.(x,z) \in R\big)$

For each of the formulae,

- provide an explanation in English;
- state whether the formula holds of the relation in part (*a*) (when the domain of x, y, and z is the set $\{\mathrm{a},\mathrm{b},\mathrm{c}\}$);
- if the formula does not hold, exhibit a relation over $\{\mathrm{a},\mathrm{b},\mathrm{c}\}$ for which the formula does hold.

[14 marks]

(*c*) Write down the introduction and elimination rules for the universal quantifier in structured proof. [3 marks]

SECTION C

5 Algorithms I

Mathematical hint: the following series converges to the indicated value if $|x| < 1$

$$\sum_{m=1}^{\infty} mx^m = \frac{x}{(1-x)^2}$$

(*a*) The binary min-heap provides a `decreaseKey()` method that, when applied to an element, decreases its key while preserving the properties of the data structure.

(*i*) Give a brief and clear description of how `decreaseKey()` works. [3 marks]

(*ii*) The standard `decreaseKey()` accepts only a positive argument. Describe an implementation that removes that restriction, that is to say a heap method that can move the key value up as well as down, as specified by the sign of its argument. [4 marks]

(*b*) Generalise the *binary* min-heap to one where nodes have not 2 but k children.

(*i*) State the two defining properties of a min-heap, one constraining the shape and one constraining the keys of the data structure, and describe how to represent a k-ary min-heap as an array. [4 marks]

(*ii*) Give a clear description of an algorithm (a simple generalisation of the well-known one for binary heaps) that takes an arbitrary n-item array and efficiently rearranges its elements to turn it into an array representing a k-ary heap. [4 marks]

(*iii*) Analyse its complexity as a function of n and k. [5 marks]

6 Algorithms I

Mathematical hint: the following series converges to the indicated value if $|x| < 1$

$$\sum_{m=1}^{\infty} mx^m = \frac{x}{(1-x)^2}$$

(*a*) You are given an unsorted array of n items of which you must return the top k, in any order. Give a clear and accurate description of two efficient algorithms for solving the problem, following the hints below, and derive their time complexity in terms of n and k.

(*i*) For this attempt, use a priority queue. [2 marks]

(*ii*) For this attempt, start by finding the k-th largest item, as when computing order statistics. Describe all the steps in detail. [8 marks]

(*b*) You are given a hash table that stores n keys. It has m slots; collisions are resolved by chaining (each hash table slot contains a pointer to the head of the corresponding chain; no elements are stored in the table itself) and no chain is longer than L elements. Each slot has an extra field giving the length of the corresponding chain.

Give a clear and accurate description of an efficient algorithm for returning a random key from the table, such that each key has equal probability of being selected, and analyse its time complexity. Assume the availability of a function `random(x)` that returns in constant time a random integer between `0` and `x`.

The time complexity of a reasonable algorithm will not exceed $O(m + L)$, but solutions awarded full marks will improve on that.

[10 marks]

SECTION D

7 Computer Fundamentals

(*a*) What is the key idea behind the *von Neumann architecture*? To what extent do modern computers conform to this architecture? [2 marks]

(*b*) Explain why modern computers contain both Dynamic RAM (DRAM) and Static RAM (SRAM). [4 marks]

(*c*) How do modern computers represent *signed integer values*? Why? [2 marks]

(*d*) In the context of assembly language programing:

(*i*) What is an *addressing mode*? [2 marks]

(*ii*) What are *pseudo instructions*? Why are they used? [2 marks]

(*iii*) What is *the stack*? What is it used for? [2 marks]

(*iv*) What is an *indirect jump*? Why would one be used? [2 marks]

(*e*) Computer A has 32 32-bit registers, while Computer B has 16 64-bit registers. Give **two** advantages that Computer A possesses over Computer B. [4 marks]

8 Floating-Point Computation

(*a*) Suppose a floating-point representation for unsigned numbers has a three-bit mantissa and a three-bit exponent. Suggest a useful encoding range by stating the minimum and maximum values representable without a hidden bit. [4 marks]

(*b*) Modify your answer to part (*a*) for when a hidden bit is used. [3 marks]

(*c*) Give roughly the smallest positive IEEE single-precision floating-point number, x, for which $\sin(x)$ is not meaningless. [3 marks]

(*d*) Describe **four** different rules for rounding a floating-point number and say which is generally used and why. [2 marks]

(*e*) Give **two** techniques for determining the number of steps used in an iteration. (Do not describe iterating until no change.) Say when one technique is preferred to the other. [4 marks]

(*f*) A scientific library uses the formula $(a+b+c)/(d+e)$ where $a \ldots e$ are floating-point. Aside from using an `IF` statement to check for division by zero, what further `IF` statement(s) should be included to ensure good precision? [4 marks]

9 Object-Oriented Programming with Java

Consider the following Java class that is intended to represent a specific day in an eight-week University term.

```
public class TermDay {
    public int day;   // The day of the week as a number 0-6
    public int week;  // The week of the term as a number 0-7
};
```

(*a*) Create a class `EncapsulatedTermDay`, which applies the principles of data encapsulation as an alternative to `TermDay`. Your modified class should throw an exception if an invalid day of the week or week number is specified. [4 marks]

(*b*) The use of two `int` variables to represent the day and the week requires 64 bits of storage. How many bits are actually required? Adapt `EncapsulatedTermDay` class to achieve the same functionality using only one member variable of a primitive type. You should justify your choice of type. [4 marks]

(*c*) Create a class `ImmutableTermDay` that is an immutable version of `TermDay`. [3 marks]

(*d*) By applying one or more appropriate design patterns and adapting `ImmutableTermDay` appropriately, show how to ensure that only one `ImmutableTermDay` object is ever created for a given day/week combination. [9 marks]

END OF PAPER