UNIVERSITY OF CAMBRIDGE

COMPUTER SCIENCE TRIPOS Part IA

NATURAL SCIENCES TRIPOS Part IA (Paper CS/1)

PSYCHOL. AND BEHAVIOURAL SCIENCES TRIPOS Part I (Paper CS 1)

Monday 30 May 2016 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

(*a*) Write brief notes on functions as values and results in ML, illustrated with the help of the functionals `map` and `exists`. What functions can we obtain from these via currying? [6 marks]

(*b*) Consider the function `zarg` defined below:

```
fun zarg f ([],    e) = e
  | zarg f (x::xs, e) = f(x, zarg f (xs,e));
```

Show that with the help of this function, it is possible to write an expression for the sum of a given list of integers. Then describe what `zarg` does in general. [4 marks]

(*c*) A polymorphic type of branching trees can be declared as follows. Note that the children of a branch node are given as a *list* of trees, and that only the leaf nodes carry labels.

```
datatype 'a vtree = Lf of 'a
                  | Br of ('a vtree) list;
```

(*i*) Write a function `flat t` that converts a given tree `t` of this type to a list of the labels (without eliminating duplicates). Your function should run in linear time in the size of the tree. [4 marks]

(*ii*) Write a function `count x t` that counts the number of times that `x` occurs as a label in `t`, but without first converting `t` to a list.
Note: Minimal credit will be given for solutions that use `flat`. [5 marks]

(*iii*) What is the type of `count`? [1 mark]

All ML code must be explained clearly and should be free of needless complexity.

2 Foundations of Computer Science

(*a*) A *prime number sieve* is an algorithm for finding all prime numbers up to a given limit n. The algorithm maintains a list, which initially holds the integers from 2 to n. The following step is then repeated: remove the head of this list, which will be a prime number, and remove all its multiples from the list. Write code for the algorithm above as an ML function of type `int -> int list`.
[4 marks]

(*b*) Consider the problem of eliminating all duplicates from a list of strings. Write code for a function of type `string list -> string list` such that the output contains the same elements as the input, possibly reordered, but where every element occurs exactly once. The worst-case performance must be better than quadratic in the length of the list. [6 marks]

(*c*) Consider the task of determining whether a given *word* (a string) can be expressed by joining together various *chunks* (non-empty strings). If the chunks are *abra*, *cad* and *hal*, then the word *abracadabra* can be expressed as *abra*|*cad*|*abra*. Note that if the available chunks are *ab*, *bra*, *cad* and *abra*, then the first two are no good for expressing *abracadabra*, and yet a solution can be found using *cad* and *abra*. The chunks can be used any number of times and in any order.

Write code for a function that takes a list of chunks along with a word, and returns a list of chunks that yield the word when concatenated. For the examples above, the result should be `["abra", "cad", "abra"]`. Exception `Fail` should be raised if no solution exists. [10 marks]

Note: You are given a function `delPrefix` for removing an initial part of a string. For example, `delPrefix ("abra", "abracadabra")` returns `"cadabra"`, but `delPrefix ("bra", "abracadabra")` raises exception `Fail`.

All ML code must be explained clearly and should be free of needless complexity. Well-known utility functions may be assumed to be available.

SECTION B

3 Object-Oriented Programming

Java generics allows an `ArrayList` object to be constrained to use a single specific type (e.g. `ArrayList<Integer>`). However, some applications require the ability to store objects of multiple unrelated types. In this question the aim is to store `Integer` objects alongside `LinkedList<Integer>` objects.

(*a*) One solution is to use `ArrayList<Object>`, since all Java objects extend `Object`. Explain why this is bad practice. [2 marks]

(*b*) Seeking to provide a solution that allows an arbitrary set of constrained types, a programmer writes an abstract `ConstrainedArray` base class. To use it, the class is extended and a specialised `void add(...)` method should be provided for each acceptable type.

```
public abstract class ConstrainedArray {
  protected ArrayList<Object> mArray =
    new ArrayList<Object>();

  public Object get(int idx) {return mArray.get(idx);}
  public int size() { return mArray.size(); }
}
```

(*i*) Show how to create a class `IntListArray` that extends this base class and accepts only `Integer` or `LinkedList<Integer>` objects. Where appropriate, objects should be copied on insertion. [4 marks]

(*ii*) Describe a sequence of events that would allow external modification of an object stored within an `IntListArray`, despite correct copying on insertion. How could this be addressed in `IntListArray`? [3 marks]

(*iii*) By adding `protected void add(Object o) {mArray.add(o);}` to the `ConstrainedArray` class, the `mArray` field can be made private. Show how this would affect your `IntListArray` class and discuss the advantages of the change from protected to private. [5 marks]

(*c*) The solutions in parts (a) and (b) both involve a `get()` method returning an `Object` reference.

(*i*) Explain why this is bad practice. [1 mark]

(*ii*) Propose an alternative solution for a constrained array of `Integer` or `LinkedList<Integer>` objects (only) that addresses this issue. [5 marks]

4 Object-Oriented Programming

(*a*) Using example Java code, distinguish between overloading, overriding and shadowing when applied to Java methods. [6 marks]

(*b*) For each of the numbered lines of code below, state whether it will compile. If it does, state and explain the output it gives when run. If it does not, explain why not. [10 marks]

```
    public static class A {
        public void print() {System.out.println("A");}
        public void printObject(A a) {System.out.println("A");}
        public static void printAll(List<A> list) {
            for (A a : list) a.print();
        }
    }

    public static class B extends A {
        public void print() {System.out.println("B");}
        public void printObject(B b) {System.out.println("B");}

        public static void main(String[] args) {
            A a = new A();
            B b = new B();
            LinkedList<A> alist = new LinkedList<A>();
            alist.add(a);
            alist.add(b);
            LinkedList<B> blist = new LinkedList<B>();
            blist.add(b);

1           ((B)a).print();
2           ((A)b).print();
3           A.printAll(alist);
4           b.printAll(blist);
5           a.printObject(b);
6           b.printObject((B)a);
7           b.printObject((A)b);
        }
    }
```

(*c*) All methods in Java are dynamic polymorphic as a design choice. Discuss the advantages and disadvantages of making dynamic polymorphism *optional* for Java methods. [4 marks]

SECTION C

5 Numerical Methods

(*a*) Consider integer division of one two's-complement binary number by another. Programming languages may vary in the result when one argument is negative. What differing conventions might they be following? [2 marks]

(*b*) Describe carefully, or give pseudocode for, the standard binary integer long-division procedure. What common fault must it guard against? [6 marks]

(*c*) Describe the iteration technique known as successive approximation by bisection. What does it have in common with standard long division? [3 marks]

(*d*) Describe carefully, or give code changes to, the standard long division algorithm from part (*b*) so that it computes two bits of result per iteration. [3 marks]

(*e*) Single-precision floating-point representation uses a sign bit, eight bits of exponent, a hidden bit and 23 bits of stored mantissa. Using a consistent encoding, similar or identical to the IEEE standard, give hexadecimal representations of the following four numbers: 1.0, 0.125, 4096.0 and -0.0. [4 marks]

(*f*) Consider different versions of an optimising compiler, each of which uses IEEE standard representation for all variables. Give two reasons why they might compile a floating-point program into code that, when run, produces differing results. [2 marks]

6 Numerical Methods

A picnicker brings hot black coffee and cold milk in two identical insulated flasks and then mixes them for his drink. His friend claims that the drink would have ended up the same temperature if he had mixed the two at home and brought one flask.

Note: The temperature of an object is the heat energy within it divided by its heat capacity. The rate of heat energy flow from a hotter to a cooler object is their temperature difference divided by their insulation resistance. When two fluids are mixed the resultant temperature is the sum of their initial temperatures weighted by their proportions.

(*a*) Give a suitable state vector for a simple, finite-difference, time domain simulation of the drink system. [3 marks]

(*b*) List the initial values and any other parameters that are needed for the simulation. Sketch pseudocode for each of the two scenarios. Assume constant ambient temperature and state any further assumptions. [7 marks]

(*c*) How would you select a fixed time step for these two simulations or should the time steps be adaptive? What accuracy might you expect to achieve? Is the choice of time step likely to affect whether the friend is proved right or wrong? [3 marks]

(*d*) Suppose the two-flask simulation were phrased in flow-matrix form. What determines how many rows the matrix would have? Is this a sensible approach to modelling the system? [4 marks]

(*e*) Why is backwards stability normally a useful property of numerical methods? Does that notion apply to this simulation? [3 marks]

SECTION D

7 Algorithms

Let a *dab* be a set of disjoint segments on the real axis.

(*a*) Write a `Dab` class in Java that supports equality testing between two dabs in linear time. Provide only the data members without any constructors or methods, and highlight any noteworthy features and invariants.
Note: You may not use any pre-made lists, resizable arrays or other library collections of any kind; use only integers, doubles, pointers, arrays and classes. You are allowed to define additional classes if necessary. [4 marks]

(*b*) Draw a records-and-pointers diagram representation of the following dab

$$\{(-2.3, -1), (10, 24.53), (2, 6)\}$$

using your class from part (*a*). [2 marks]

(*c*) Given a set S of dabs, find an algorithm that returns a dab of maximum cardinality containing only segments from the dabs in S. The algorithm should run in $O(n^2)$ time, where n is the total number of segments of all dabs contained in S.

(*i*) Clearly describe and explain your algorithm. [4 marks]

(*ii*) Describe your algorithm in a few lines of pseudocode. [4 marks]

(*iii*) Prove that your algorithm is correct. [4 marks]

(*iv*) Derive the asymptotic running time of your algorithm. [2 marks]

8 Algorithms

(*a*) Transform the following recurrence

$$f(x) = f(\sqrt{x}) + c$$

into a closed-form expression for the function f (that is, an expression that does not contain f). Having done that, give the asymptotic complexity of f using big-O notation. [4 marks]

(*b*) (*i*) Explain the programming technique known as *memoization*, detailing the cases to which it applies. [4 marks]

(*ii*) In a few lines of pseudocode, write a memoized recursive function to compute the ith Fibonacci number $F(i)$, with $i \in \mathbb{N} \setminus \{0\}$. Recall that $F(1) = 1, F(2) = 1$,... [4 marks]

(*c*) Computing a recursive function f on arrays, when called on an array of size n, results in 2^n recursive calls to f. After memoizing f, on an array of a specific size n_0 we observe that about 90% of the calls to f return a memoized result rather than invoking f recursively. Is either of the following statements correct? Justify your answers.

(*i*) "The number of recursive calls goes down by a factor of ten; so it will take 1/10 of the time it used to, that is, it will run 10 times faster." [2 marks]

(*ii*) "Previously, the function did 2^n recursive calls. Now it does $0.9 \cdot c_1 + 0.1 \cdot 2^n$ recursive calls. That is still $O(2^n)$, so the asymptotic complexity of the function is still the same (even after memoization)." [2 marks]

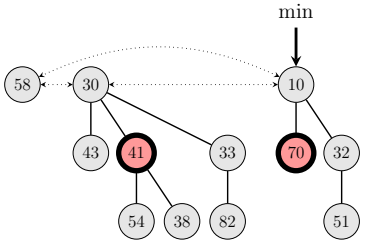
(*d*) Some implementations of the Quicksort algorithm select the pivot at random, rather than taking the last entry in the input array.

(*i*) Discuss the advantages and disadvantages of such a choice. [1 mark]

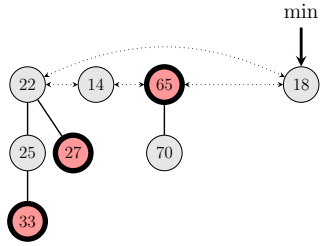
(*ii*) How would you construct an input to trigger quadratic running time for this randomised Quicksort, without having access to the state of the random number generator? [3 marks]

9 Algorithms

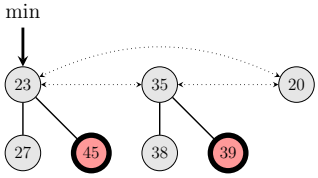
(a) State whether each of the following six forests can be generated from an empty Fibonacci Heap using the standard operations. If it cannot be generated, clearly state the condition or property of Fibonacci Heaps that is violated.
Note: Marked nodes are in boldface. [9 marks]



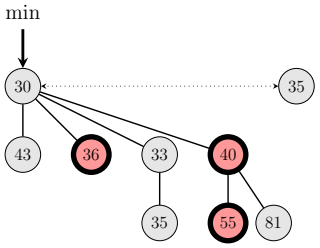
(i)



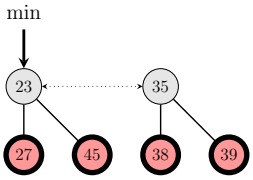
(ii)



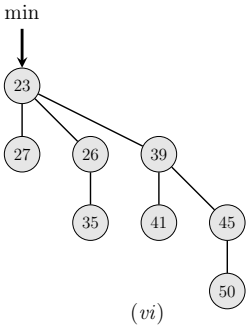
(iii)



(iv)



(v)



(vi)

(b) Consider the operations Insert, Extract-Min, and Decrease-Key in a Fibonacci Heap. Answer the following for each of the three operations.

(i) What are the worst-case actual costs? [3 marks]

(ii) What are the worst-case amortized costs? [3 marks]

(c) Consider the following modification to Fibonacci Heaps. Instead of marking a node once it has lost its first child, we mark a node once it has lost two of its children. How would you adjust the analysis of the maximum degree $d(n)$? [5 marks]

10 Algorithms

(*a*) We consider the minimum spanning tree problem. For each of the following three scenarios, which algorithm would you use for efficiency and what would its running time be?

(*i*) Graph with V vertices, $E = \Theta(V^{3/2})$ edges with arbitrary weights in $\mathbb{R}$.

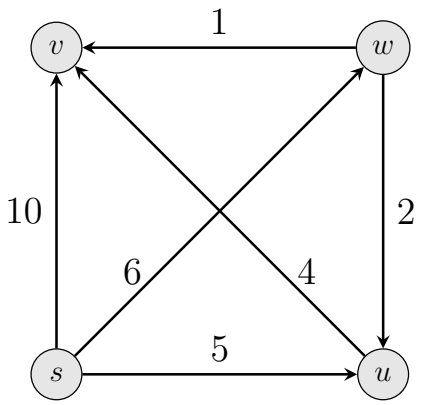
(*ii*) Graph with V vertices, $E = \Theta(V(\log V)^2)$, where edges have already been sorted according to their weights.

(*iii*) Graph with V vertices, $E = \Theta(V)$ edges with arbitrary weights in $\{0, 1, \ldots, \lfloor \log V \rfloor\}$.

[6 marks]

(*b*) What are the advantages and disadvantages of the Bellman-Ford Algorithm in comparison to Dijkstra's Algorithm? [4 marks]

(*c*) For the graph with source vertex s below, perform Dijkstra's Algorithm. It is sufficient to state the order in which the four vertices are extracted from the priority queue as well as their actual distances from the source s.



[6 marks]

(*d*) Suppose that we are given a weighted, directed graph $G = (V, E)$ in which edges that leave the source vertex s may have negative weights, but all other edge weights are non-negative. Does Dijkstra's Algorithm correctly find the shortest path distances from s? Either give a proof of correctness or provide a counter-example. [4 marks]

END OF PAPER