COMPUTER SCIENCE TRIPOS Part IA

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

Monday 5 June 2017 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 one-person game (such as Rubik's cube, or peg solitaire) has a finite number of possible *states*, some of which count as *winning*. A *move* is a step from one state to another. From each given state, the player can choose from a set of (zero or more) possible next moves. We call a state *winnable* if a winning state can be reached from it in zero or more moves.

For simplicity, assume that states are coded as integers. Also assume that we are given functions `winning(x)` returning true or false and `next(x)` returning the list of states that can be reached in one move from state `x`.

(a) The following code is an attempt to implement the notion of winnable:

```
fun exists p []      = false
  | exists p (x::xs) = p x orelse exists p xs;

fun winnable x = winning x orelse exists winnable (next x);
```

Briefly explain how this code works. Also describe its main limitation: how it can fail to find a winning state that is only a few moves away. Illustrate this point by giving specific definitions of `winning` and `next`. [5 marks]

(b) Modify the code above to yield the function `winpath x`, which returns the list of states from `x` to the winning state found or, alternatively, the empty list to indicate that no winning state was found. [4 marks]

(c) Sometimes we are only interested in a winnable state if it is only a few moves away from the current state. Modify your solution from part (b) to obtain the function `bounded_winpath n x`, which looks for winning states that are at most `n` moves away from `x`. [3 marks]

(d) Use your solution from part (c) to obtain the function `new_winpath x`, which has the same objective as `winpath x`, but without the limitation mentioned in part (a). Briefly explain why the limitation no longer applies and the price that has been paid for this. [5 marks]

(e) Briefly outline an alternative approach to correcting the limitation mentioned in part (a), using the notion of a queue. What are the advantages and drawbacks of this approach? [3 marks]

For full credit, code should be concise and clear. Exceptions may be useful in this but are not required.

2 Foundations of Computer Science

(a) Define an ML datatype for infinite lists, without the possibility of finite lists. Briefly illustrate programming techniques for your datatype by declaring

(i) a recursive functional (analogous to map for ordinary lists) that applies a given function to every element of an infinite list.

(ii) a function for generating infinite lists of the form x, $f(x)$, $f(f(x))$, ... for any given f and x.

[6 marks]

(b) Briefly explain why the function analogous to append (@) for ordinary lists is of no value for your infinite list data type. Code a function to combine two arbitrary infinite lists, yielding a result that includes the elements of both lists.
[3 marks]

(c) Use the functions declared in your previous solutions to express an infinite list consisting of all numbers of the form $5^i \times 7^j \times 9^k$ for integers i, j, $k \geq 0$.
[3 marks]

(d) The list $[1, 5, 7, 25, 9, 35, 35, \ldots]$ is a legitimate solution to part (c) above, but note that the integers are out of order. Code a function to merge two ordered infinite lists, and use that to modify your previous solution to yield the same set of numbers but in strictly increasing order. Briefly comment, with justification, on whether merge sort for ordinary lists can be generalised to infinite lists.
[8 marks]

For full credit, code should be concise and clear.

SECTION B

3 Object-Oriented Programming

An online retailer uses custom Java software to manage their inventory and sales.

(*a*) Each product sold is represented using an immutable `Product` object. Explain what is meant by immutable, how immutability is typically achieved in Java and the advantages of using immutable objects in general. [4 marks]

(*b*) `Product` objects are requested through a `Product getProduct(long code)` method, which returns a reference to a `Product` object given a valid product code. Product information is often re-requested as customers make their selections so the 10,000 most recently *accessed* `Product` objects are cached in memory. Uncached `Product` objects are created with information retrieved from a database when requested.

(*i*) The cache uses a `java.util.HashMap` and a custom implementation of a doubly-linked list. The list keeps an ordering over the `Product` objects where more recently used objects are at the front. The `HashMap` provides fast lookup into the list. Show that this scheme gives a constant (O(1)) running cost for `getProduct()`, ignoring the cost of the database lookup. [3 marks]

(*ii*) Create a class `Store` that implements the cache as described. You need only define `getProduct` and any state or definitions it needs. All other state and methods can be ignored. You may assume the existence of a method `loadFromDatabase(long code)` that will create a `Product` object for product code `code` or return `null` if the code is invalid, and that a `Product` object has a `long getProductCode()` method that returns its product code. [10 marks]

(*c*) A customer's basket of items, represented by a class `Basket`, can be viewed as a list of products. Therefore `Basket` might extend `LinkedList<Product>`. Compare this approach to a `Basket` that contains ("has-a") `LinkedList<Product>` instead. [3 marks]

4 Object-Oriented Programming

(*a*) Give four advantages of Java's checked exceptions over return values for error indication. [4 marks]

(*b*) Comment on the appropriate use of Java's checked exceptions within `public`, `protected` and `private` methods. [6 marks]

(*c*) Consider a method that can encounter at least two errors (*Error1* and *Error2*). Compare and contrast the following approaches to providing exceptions for these errors.

(*i*) `throw new MethodError()`, where `MethodError` is a direct subclass of `Exception`.

(*ii*) `throw new Exception()` for both errors.

(*iii*) `throw new MethodError(errortype)`, where `MethodError` directly subclasses `Exception` and contains state recording which error occurred (initialised by parameter `errortype`).

(*iv*) `throw new Error1()` and `throw new Error2()`, where `Error1` and `Error2` directly subclass `MethodException`, which directly subclasses `Exception`.

(*v*) `throw new Exception("Error1")` and
`throw new Exception("Error2")`.

(*vi*) `throw new Error1()` and `throw new Error2()`, where the classes `Error1` and `Error2` directly subclass `Exception`.

[10 marks]

SECTION C

5 Numerical Methods

A programmer is asked to write a function that raises a single-precision number to the power of 2.4. His manager advises him to take the following approach involving squaring and fifth roots, and to use Newton Raphson for the fifth root step.

$$f(x) = x^{2.4} = x^2 \times \sqrt[5]{x^2}$$

(*a*) State why the above identity holds. [1 mark]

(*b*) Sketch an implementation of this approach coded in Java or ML. Use the four standard arithmetic operators but do not use any other built-in maths operators or libraries. You must include an iteration termination strategy. [8 marks]

(*c*) Assuming values of x of around unity, and stating any other assumptions you make, give the worst-case error expected in your result assuming the input x is perfectly-representable. [*Note:* macheps for single precision $\epsilon = 1.19 \times 10^{-7}$ but you may give your answer in terms of ϵ.] [3 marks]

(*d*) What worst-case error arises when the input value was approximated by the nearest representable value and so is already out by as much as $\pm\epsilon/2$? [2 marks]

(*e*) A colleague suggests it would be better to simply use the logarithm and exponential functions in the standard library. Another says you should just use a suitable polynomial evaluation (i.e. a power series) for the complete problem. Discuss whether either of these might be better or worse than the original suggestion in terms of performance and accuracy. [6 marks]

6 Numerical Methods

A seismic probe bores itself into the seabed, going as deep as it can before running out of fuel. This takes about five minutes. It rotates its spiral drill head at rate $R(t)$ that follows a pre-programmed profile. Its downwards velocity $\dot{y}$ is proportional to R and to the square-root of the probe's weight w, and is given by

$$\dot{y} = 3.6 \ \times \ R(t) \ \times \ \sqrt{w}.$$

The weight of the probe is the weight of its fuel, which starts at 1500 plus its own intrinsic weight, which is fixed at 35. The fuel weight decreases at a rate of 1.2 $R(t)$.

(*a*) Give the state vector for a forwards FDTD simulation of this system using Euler's Method. [1 mark]

(*b*) Assuming a function is provided that returns $R(t)$, give a program that uses Euler's Method (a straightforward, forward finite-difference simulation) to determine the depth achieved. [4 marks]

(*c*) Describe the errors you might expect if you chose a time step that was inordinately small or inordinately large. What is the maximum time step for Euler's method to remain stable in this system? Suggest, with justification, a suitable time step. [6 marks]

(*d*) Recall that a backwards stencil uses the values at the end of the timestep to determine the rate of change during that timestep. When is this usable and useful in general? Is it sensible for this application? Give modified code that implements the backwards stencil method. [6 marks]

(*e*) For the probe to drill as deep as possible, should $R(t)$ generally start small and grow larger? Justify your answer. [3 marks]

SECTION D

7 Algorithms

This question is about Binary Search Trees (BSTs) and Red-Black Trees (RBTs).

(a) Using a diagram, explain what a BST rotation is and its purpose. [3 marks]

(b) Consider the following buggy pseudocode.

```
0  def mystery(x):
1      y = x.r
2      x.r = y.l
3      if y.l != null:
4          y.l.p = x
5      x.p = y.p
6      if x == x.p.l:
7          x.p.l = y
8      else:
9          x.p.r = y
10     y.l = x
```

(i) Explain what it intends to do, give it a meaningful name, describe all the identifiers used (`x`, `y`, `r`, `l`, `p`) and the (intended) precondition and postcondition of the routine. [4 marks]

(ii) Identify, explain and fix the bugs, one by one, referring to a diagram if useful. Finally, give a fully corrected version of the code. [8 marks]

(c) State, with a proof or counterexample as appropriate, whether each of the following statements is true or false.

(i) In an RBT with more than one node, at least one node is red. [2 marks]

(ii) In a BST with n nodes, exactly $n - 1$ rotations are possible. [3 marks]

8 Algorithms

This question is about hash tables.

(*a*) Briefly explain *hash functions*, *hash tables* and *collisions*. [3 marks]

(*b*) Explain the *open addressing* strategy of collision resolution and the term *probing sequence* used in that context. [3 marks]

(*c*) Explain *quadratic probing* and its advantages and disadvantages. [*Hint:* refer to primary and secondary clustering.] [3 marks]

(*d*) Give a general mathematical expression for the probing function $p(k, i)$ used in quadratic probing. The expression should yield a 0-based index into the table, referencing the key k, the probe number i, the hash function h, the table size m and the constants c_1 and c_2. [3 marks]

(*e*) Does the following pseudocode implement a form of quadratic probing? If so, derive values for c_1 and c_2 in the equation you produced for part (*d*). If not, prove it doesn't. In either case, clearly justify your reasoning. [8 marks]

```
def get(k):
    j = h(k)
    i = 0
    while true:
        if T[j].key == null: raise NotFound
        if T[j].key == k: return T[j].payload
        i = i+1
        if i == m: raise NotFound
        j = (i+j) mod m
```

9 Algorithms

We wish to store a dynamic collection of records, each of the form `{timestamp, value}`, where `value` is a real number. The collection should support the operations `append_newer(t,v)` to add a new record (which we can assume has a larger timestamp than any existing record), `pop_oldest()` to remove the oldest record, and `get_oldest()` to inspect the oldest without removing it.

(*a*) Define the `Queue` abstract data type. Describe an implementation using a linked list. Explain how to use it for this dynamic collection of records. [3 marks]

The collection should also support `get_max()`, which returns a pointer to the record with the highest `value` in the collection. Ties may be broken arbitrarily.

(*b*) A simple implementation of `get_max()` simply scans through the entire list. What is the worst-case cost, given the number n of items in the collection? [1 mark]

(*c*) An engineer friend suggests keeping a pointer `maxrecord` to the record with the largest value so that the entire list only need be rescanned when the item pointed to by `maxrecord` is removed. Give an example to show that n operations could take $\Omega(n^2)$ time. [3 marks]

(*d*) Explain the terms *amortized cost* and *potential method*. Explain the relationship between aggregate true costs and aggregate amortized costs. [4 marks]

(*e*) Devise an implementation in which all operations have $O(1)$ amortized cost, and use the potential method to justify your answer. Illustrate what happens when we start with a list of values $[5, 8, 3, 6, 2]$ where 5 is oldest and 2 is newest, and then append a newer record with value 7. [*Hint:* Where is the largest item newer than `maxrecord`, and the largest item newer than this, and so on?] [9 marks]

10 Algorithms

(*a*) Consider a directed acyclic graph with V vertices and E edges.

(*i*) What is meant by *a total order on the vertices consistent with the edges*? [2 marks]

(*ii*) Describe an $O(E+V)$ algorithm to compute such a total order. [3 marks]

(*b*) Consider a directed graph with non-negative edge costs and with a given start vertex s.

(*i*) Dijkstra's algorithm computes distances from s to every other vertex. Give psuedocode for Dijkstra's algorithm. [4 marks]

(*ii*) Dijkstra's algorithm can be implemented using a Fibonacci heap. State the complexity of using this implementation. Justify your answer carefully. [*Note:* Your answer should include mention of amortized costs.] [4 marks]

(*c*) Consider a directed acyclic graph with non-negative edge costs and with a given start vertex s.

(*i*) Devise an algorithm to compute distances from s in $O(E+V)$ time. Justify why your algorithm is correct. [4 marks]

(*ii*) Explain, with an example, why Dijkstra's algorithm might take $\Omega(V \log V)$ time. [3 marks]

END OF PAPER