CST0
COMPUTER SCIENCE TRIPOS Part IA

Monday 6 June 2022 14:00 to 17:00 BST

COMPUTER SCIENCE Paper 1

*Answer **one** question from each of Sections A, B, C, D, and E.*

*Submit each question answer in a **separate** PDF. As the file name, use your candidate number, paper and question number (e.g., `1234A-p1-q6.pdf`). Also write your candidate number, paper and question number at the start of each PDF.*

You must follow the official form and conduct instructions for this online examination

SECTION A

1 Foundations of Computer Science

A single-player game requires you to guess the letters of a *target* word within a limited number of tries. Each try the player guesses a single letter c and integer position i (starting from 0). The game responds: `Green` if c appears in the position i in the target, `Amber` if c is present elsewhere within the target, or `Black` if c is not present in the target. For simplicity, repeated letters are not allowed in the target.

```
type word = char list
type guess = char * int
type guesses = guess list
val prune_guesses : guesses -> guesses
```

The `word` type is used to list the correct letters in order, `guess` is a single guess (the letter and its position within the target `word`) and `guesses` is used to list the guesses in order with the most recent first. You may assume that a `prune_guesses` function is available that removes all but the most recent guess for each character.

(a) Define two functions `mapi` and `lookfor` with the following types:

```
val mapi : (int -> 'a -> 'b) -> 'a list -> 'b list
val lookfor : 'a -> ('b * 'a) list -> 'b option
```

The call `mapi f l` maps over every element of `l` using a function `f` that accepts the position of the current list element along with the element. For example, `mapi f ["a";"b";"c"] = [ f 0 "a"; f 1 "b"; f 2 "c" ]`.

The call `lookfor y l` searches `l` for a pair where the second component is equal to `y`, and returns the first component in the pair if found. [4 marks]

(b) Using the earlier definitions, or otherwise, write a function

```
val respond : word -> guesses -> responses
```

that takes a target word and a list of guesses and returns feedback about the game progress to the player. You should define a `responses` type by referring to the game rules. [8 marks]

(c) (i) Define a function `create_game: word -> (guess -> responses)` that returns a function g which can be used to imperatively play an independent game: any such g can be called for at most six tries after which it raises an `Out_of_turns` exception. [6 marks]

(ii) Illustrate with a brief example how you would use such a function g to make some tries, showing the three possible response types. [2 marks]

2 Foundations of Computer Science

One way to represent sets of integers is as lists of intervals:

```
type intset = (int * int) list
```

For example, $\{1, 2, 3, 9, 10, 11, 12\}$ can be represented as `[(1,3);(9,12)]`, the union of the intervals [1..3] and [9..12].

(a) Each set of integers has many different interval list representations. An interval list (`intset`) is in *standard form* if it is an ascending sequence of non-empty intervals that cannot be merged.

(i) Write a function that tests whether an `intset` is in standard form:

```
val is_standard : intset -> bool
```

[4 marks]

(ii) Write a function that adds an interval to an `intset` in standard form, producing a new `intset` in standard form:

```
val add_interval : (int * int) -> intset -> intset
```

[4 marks]

(iii) Write a function that converts an `intset` to standard form:

```
val standardize : intset -> intset
```

[2 marks]

(iv) Write a function to test whether two `intset` values represent the same set:

```
val equal : intset -> intset -> bool
```

[2 marks]

(b) Write a function that computes the intersection of integer sets:

```
val inter : intset -> intset -> intset
```

You may assume that the arguments to `inter` are in standard form. [8 marks]

SECTION B

3 Object-Oriented Programming

(*a*) Give three advantages and one disadvantage of immutable classes. [4 marks]

(*b*) A programmer has created an `AssetLocation` class to represent the location of company assets. Some assets are mobile and their location must be updated regularly (e.g. vehicles); others are static and will never be updated (e.g. warehouses).

The class contains a `String` describing the asset, an `int` recording a unique identifier, and two `double` values to represent valid latitude ($-90° \leq \phi \leq 90°$) and longitude ($-180° < \theta \leq 180°$) values, respectively. All fields are initially mutable and set by the constructor.

Write Java code that implements `AssetLocation` as described. [5 marks]

(*c*) The programmer wishes to make the objects representing static assets immutable. They make the class *optionally* immutable using a parameter passed into the constructor.

(*i*) Write a modified `AssetLocation` class that implements the behaviour as described. [3 marks]

(*ii*) Explain why this is not a good solution. [3 marks]

(*iii*) Propose a better structure for the class, and explain your design choices. [5 marks]

4 Object-Oriented Programming

(*a*) Explain the notion and value of type checking. Illustrate your answer using generic types in Java. [3 marks]

(*b*) Explain the notion of type erasure. Why is it used in Java? [3 marks]

(*c*) For each of the statements below, state whether they are true or false, and explain why.

(*i*) `Integer` can be cast to `Number`. [1 mark]

(*ii*) `Set<Integer>` can be cast to `Set<Number>`. [2 marks]

(*iii*) `Set<Integer>` can be cast to `Set<? extends Number>`. [2 marks]

(*iv*) `Set<Integer>` can be cast to `Set<?>`. [2 marks]

(*v*) `Set<Set<Integer>>` can be cast to `Set<Set<?>>` [2 marks]

(*d*) Given a generic class `TreeSet<T>`, explain the difference between:

(*i*) `new TreeSet()`

(*ii*) `new TreeSet<Object>()`

(*iii*) `new TreeSet<>()`

When can each of these Java expressions be used and are there any advantages of using one over another? [5 marks]

SECTION C

5 Introduction to Probability

A student is taking a multiple-choice exam consisting of n questions. Assume for each question, the student either knows the correct answer or guesses one of the four available answers randomly. Further, assume that for each question, the student knows the correct answer with probability $p = 0.6$, and these events are independent across different questions.

(*a*) Let Z be the random variable counting the correct answers for the n questions given by the student. What is the distribution of Z? Also state its expectation and variance. [4 marks]

(*b*) Assuming $n \to \infty$, how could you approximate the distribution of Z in order to estimate $\mathbf{P}[Z \geq x]$ for some $x \geq 0$? [4 marks]

(*c*) Given that the student answers a question correctly, what is the probability that the student actually knows the answer? [4 marks]

(*d*) Suppose we know that the student will answer all n questions of the exam correctly with probability 0.343. What is the number of questions n on the exam? [2 marks]

Consider now a refined probabilistic model with $n = 2$ questions. For $i \in \{1, 2\}$, let X_i be an indicator random variable which is 1 iff question i is answered correctly by the student. We know $\mathbf{P}[X_1 = 1 \mid X_2 = 0] = 0.4$, $\mathbf{P}[X_2 = 1 \mid X_1 = 1] = 0.8$ and $\mathbf{P}[X_1 = 1] = 0.5$.

(*e*) Compute the joint distribution of (X_1, X_2). [4 marks]

(*f*) Are X_1 and X_2 independent? [2 marks]

6 Introduction to Probability

(a) A laptop is expected to run for X number of hours from new until it breaks down. Let X be an exponential random variable with $\lambda = \frac{1}{25000} = 4 \cdot 10^{-5}$.

(i) You are given a laptop that has already been used for $4000 = 4 \cdot 10^3$ hours. What is the probability that you will be able to use it for another $10000 = 10^4$ hours? [3 marks]

(ii) The laptop keeps getting passed on, from one owner to another. Each owner uses this laptop for $10000 = 10^4$ hours. This continues until the laptop breaks down. What is the probability that the laptop breaks down for the n^{th} owner, $n \geq 1$? [4 marks]

(iii) Let R be a random variable for the number of laptop owners until it breaks down. Show that R is a geometric random variable, and give the value of its parameter p. [3 marks]

(iv) Consider now two different laptops with lifetimes X_1, X_2, which are two independent exponential random variables with rates $\lambda_1 = 4 \cdot 10^{-5}$ and $\lambda_2 = 8 \cdot 10^{-5}$. What is the expected time until the first of the two laptops breaks down? [4 marks]

(b) We know that a laptop battery breaks down from new after T charging cycles, where T is a geometric random variable with parameter $p \in (0, 1)$.

(i) What is $\mathbb{E}[T]$? Find an unbiased estimator for $\mathbb{E}[T]$. [2 marks]

(ii) Find an unbiased estimator for p and interpret the result. [4 marks]

SECTION D

7 Algorithms 1

Consider binary trees, represented as `BinTree` objects and whose nodes are represented as `Node` objects. Both have the expected structure: the `BinTree` object points to a `root Node` object. Each `Node` object has `left`, `right` and `parent` pointers to its left child, right child and parent nodes respectively; these can be null. Additionally, each node has a `container` pointer to the (unique) `BinTree` that contains it, and has `key` and `value` fields. There are no duplicate keys within a `BinTree`.

Now assume functions: nodes(T) which gives the set of all nodes in `BinTree` T, and descendants(n) which gives the set of nodes reachable by following zero-or-more `left` and `right` links from `Node` n.

Let `BT1` and `BT2` be properties of a `BinTree` T, defined by:

$$\texttt{BT1}(T) \iff$$
$$\forall n \in \text{nodes}(T) : n.\mathit{left} \neq \text{null} \Rightarrow n.\mathit{left}.\mathit{key} < n.\mathit{key} \quad (1)$$
$$\wedge \quad \forall n \in \text{nodes}(T) : n.\mathit{right} \neq \text{null} \Rightarrow n.\mathit{key} < n.\mathit{right}.\mathit{key}. \quad (2)$$

$$\texttt{BT2}(T) \iff$$
$$\forall n \in \text{nodes}(T), \forall m \in \text{descendants}(n.\mathit{left}) : m.\mathit{key} < n.\mathit{key} \quad (3)$$
$$\wedge \quad \forall n \in \text{nodes}(T), \forall q \in \text{descendants}(n.\mathit{right}) : n.\mathit{key} < q.\mathit{key}. \quad (4)$$

(a) State whether each of the following statements is true or false, justifying your statement with a proof or counterexample.

(i) $\forall$ `BinTree` T : `BT1`(T) $\Rightarrow$ `BT2`(T). [2 marks]

(ii) $\forall$ `BinTree` T : `BT2`(T) $\Rightarrow$ `BT1`(T). [2 marks]

(b) Write neat and well-commented pseudocode for a `void` method `deleteRoot` of class `BinTree`. When given a `BinTree` T that satisfies `BT2`(T), the method must delete the root note and rearrange T so that it continues to satisfy `BT2`(T). It is an explicit requirement that you delete the root node of T, as opposed to deleting some other node and copying its key and value into the former root. [*Hint:* sketching diagrams of the various cases and referring to them in comments is not required but may help you write correct pseudocode.]

[7 marks for correctness + 7 marks for clarity]

(c) What undesirable consequences might ensue if a programmer violated the explicit requirement specified in Part (b)? [2 marks]

8 Algorithms 1

Given an array a containing n items to be sorted, a bottom-up implementation of mergesort performs, non-recursively, several passes on a.

(a) Derive $p(n)$, the number of passes performed. [1 mark]

(b) Derive $m(n, i)$, the number of merge operations performed in pass i, where passes are numbered starting from 0 and ending at $p(n) - 1$. [2 marks]

(c) A programmer has (correctly) read that an array a of n elements can be sorted with bottom-up mergesort using scratch workspace of size $\lceil n/2 \rceil$ elements. The programmer decides to implement this by requiring the caller to arrange that a starts with n cells containing the values to be sorted, followed by $\lceil n/2 \rceil$ cells to be used as workspace, and produces the following pseudocode:

```
0  def bums(a, n):
1      """Bottom-up-merge-sort a[:n], using a[n:] as scratch space."""
2      assert len(a[n:]) >= n/2 # NB: here n/2 is not integer for odd n
3      s = 1 # Size of the chunks to be merged in this pass
4      for pass between 0 included and p(n) excluded:
5          for pair between 0 included and m(n, pass) excluded:
6              copy a[s*pair:s*(pair+1)] to a[n:n+s]
7              srcA = n
8              maxA = n + s
9              srcB = s * (pair+1)
10             maxB = max(s * (pair+2), n)
11             dst = s * pair
12             while (srcA < maxA) or (srcB < maxB):
13                 if a[srcA] < a[srcB]:
14                     a[dst++] = a[srcA++]
15                 else:
16                     a[dst++] = a[srcB++]
17         s = 2 * s
```

This pseudocode contains three serious bugs. For each of them:

(i) Explain the bug clearly, focusing on the difference between programmer's intention and the code as written; then suggest how to fix it (no pseudocode is required). [3 marks each]

(ii) In the spirit of unit testing, exhibit a simple input pair (a and n) that triggers *that* bug but neither of the others, contrasting intended and actual behaviour. [2 marks each]

(d) Assuming the bugs in Part (c) are corrected, is this bottom-up mergesort implementation stable? Give reasons. [2 marks]

SECTION E

9 Algorithms 2

In Part II of the Computer Science Tripos at Cambridge, students take two Units of Assessment (UA). Some UAs have a maximum capacity, set by the lecturer. Students are asked to shortlist three or more UAs, and the department tries to find class rosters such that each student is assigned two of the UAs they shortlisted.

(a) Give an efficient algorithm for finding such rosters, if they can be found. [5 marks]

(b) State an appropriate correctness property, and prove it. [7 marks]

(c) Derive the running time, in big-O notation, as a function of the number of students. Treat the number of courses as fixed. [3 marks]

Oxbridge Academy has a similar programme, but larger: students are required to take three UAs, and they shortlist at least four. Some UAs are in Michaelmas term, others in Lent, and the Academy wishes to create class rosters such that no student has all three of their UAs in a single term.

(d) Modify your algorithm to accommodate this requirement. [5 marks]

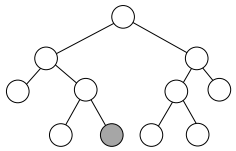
[You may use standard algorithms and results about them provided you state them clearly.]

10 Algorithms 2

Consider a Dictionary whose keys belong to a totally ordered set, and whose values are real numbers. We would like to implement an additional operation: `partialsum(`k`,`k'`)` which sums all values whose key ℓ satisfies $k \leq \ell < k'$.

We can implement this Dictionary using a balanced binary search tree, and implement `partialsum` by first searching for k then calling `successor` until we reach a key $\ell \geq k'$ or we run out of keys. (The `successor` function, when applied to a node in the tree whose key is k, returns the node with the smallest key that is $> k$, if one exists.)

We can analyse the cost of `partialsum` by treating it as a sequence of operations: one search, then one or more calls to `successor`. We can analyse the cost of this sequence using the potential method.



(*a*) In the tree shown above, label nodes by the order in which they are visited by successive calls to `successor`, starting from the shaded node. [2 marks]

(*b*) Give pseudocode for the `successor` function. Show that the worst-case cost of `successor` is $\Omega(\log n)$, where n is the number of items in the tree. [5 marks]

(*c*) Consider the function

$$\Phi(k) = 2r_k + D - d_k$$

where D is the depth of the tree, r_k is the number of right-child steps on a path from root to the node with key k, and d_k is depth of that node. Augment Φ by defining its value at an 'initial empty' state, which you should define. Explain why your augmented function is a potential function. [3 marks]

(*d*) Show that `partialsum` is $O(m + \log n)$, where n is the number of items in the tree and m is the number of calls to `successor`. Explain your reasoning. [10 marks]

END OF PAPER