COMPUTER SCIENCE TRIPOS Part IB

Monday 1 June 2020 1.30 to 4.30

COMPUTER SCIENCE Paper 4

*Answer **five** questions: **up to four** questions from Section A, and **at least one** question from Section B.*

*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 Programming in C

Consider expressions represented using the following ML datatype:

```
datatype exp = Var of string | Neg of exp | Divide of exp * exp
```

(*a*) Using at least one `union`, define a type or types in the C language for conveniently storing such expressions. [5 marks]

(*b*) Give efficient C code that checks whether two structures represent identical expressions. (Do not consider whether they might evaluate to the same result.) Explain how much of the input expressions is explored when they differ. [4 marks]

(*c*) Given that a lot of expressions are to be rapidly generated and discarded, what considerations apply to storage management? Define and discuss at least 3 different approaches to storage management. [5 marks]

(*d*) The substitution operation for an expression replaces all occurrences of one variable with another variable. Given that sub-expression trees are commonly shared over numerous expressions, explain a problem that could arise in the substitution operation. Explain the details of a solution to the problem by giving code or otherwise. [6 marks]

2 Programming in C and C++

(*a*) Arrays in C are accessed using square bracket notation.

(*i*) What are the advantages and disadvantages of array bounds checking?

(*ii*) Define with examples an array access and a pointer de-reference in the C language explaining the underlying equivalents.

(*iii*) State all similarities between array access and pointer dereference and comment on interactions with possible bounds checks.

[5 marks]

(*b*) What advantage is gained from allowing a given C program to vary in execution behaviour from one computer architecture to another? Give three common example variations. What is the disadvantage of variation? [5 marks]

(*c*) Give two ways in which C++ templates differ from Java Generics, other than mere syntactic differences. [4 marks]

(*d*) Giving a suitable example, explain the effect in C++ of qualifying a member function (method) with `virtual`. [3 marks]

(*e*) Recode the following Java code in C++. Minor syntactic errors will not be penalised.

```
class Foo
{   final int[] v; final int s;
    public Foo(int n) { s = n; v = new int[n]; }
    // In Java garbage collection de-allocates arrays
    // appearing in no-longer-used instances of Foo.
    // In C++ an alternative solution is required.
}
```

[3 marks]

3 Compiler Construction

(a) In the context of the compilation of functions, what is a *closure*? [2 marks]

(b) The front-end of our Slang compiler eliminates let-bindings by replacing the code

```
let x e1 in e2 end
```

with the code

```
(fun x -> e2 end) e1
```

Apply this transformation to the following Slang code.

```
let f(x) =
    let x1 = e1
    in let x2 = e2
       in e3 end
    end
in
    e
end
```

[3 marks]

(c) Describe the structure of the Jargon code generated from the Slang in your answer to Part (b). (Don't worry about getting the syntax exactly right.) [6 marks]

(d) Consider the Jargon code generated in Part (c). Suppose the function `f` is called with the value v somewhere in the code generated from the expression `e`. Describe what happens at runtime when `f(v)` is executed. In particular, describe the closures that exist in the heap and how they are used to evaluate `f(v)`. [4 marks]

(e) Describe a better way of compiling let-bindings such as those associated with `x1` and `x2` in the code above. Rather than creating closures, the idea is to include these "local variables" in the stack frame for `f`. Explain in detail how this might be done. [5 marks]

4 Compiler Construction

Consider the following Context Free Grammar

$$S \rightarrow Aa \mid BAb$$

$$A \rightarrow BB \mid c$$

$$B \rightarrow Sd \mid e$$

where $\{a,\ b, c,\ d,\ e\}$ is the set of terminal symbols.

(a) Give a right-most derivation of *ecadeb*. [1 mark]

(b) Give a left-most derivation of *ecadeb*. [1 mark]

(c) Compute FIRST and FOLLOW for this grammar. Show your work. [6 marks]

(d) Is the grammar LL(1)? Justify your answer. [4 marks]

(e) Is the grammar SLR(1)? Justify your answer. [8 marks]

5 Further Java

A concurrent queue allows two or more Java threads to add items to, or remove items from, a shared buffer. Items are stored in the buffer in a strict first-in-first-out order. The buffer in this question has a fixed size and therefore once the buffer is full, any attempt to add items to the buffer should block until enough items are removed that there is space to store the additional items. If the buffer is empty, any attempt to remove items from the buffer should block until an item becomes available.

A novice Java programmer implements a concurrent queue as follows:

```
public class IncorrectConcurrentFixedSizeQueue {
  private int[] buffer = new int[10];
  private int front, back, count;
  public void put(int val) throws Exception {
      while (count++ >= 10)
        Thread.sleep(1000);
      buffer[front++ % 10] = val;
  }
  public int get() {
    count--;
    return buffer[back++ % 10];
  }
}
```

(*a*) Describe the intent of the variables `buffer`, `count`, `front` and `back`. [4 marks]

(*b*) Describe, with justification, three problems with the above implemention which means it does not adhere to the specification of a concurrent queue. [6 marks]

(*c*) Write a new implementation of a *generic* concurrent queue which stores items of type `T`. The constructor should accept an integer to specify the fixed size of the buffer. [8 marks]

(*d*) Two Java threads form a pipeline where the output of the first thread is used as an input into the second. A concurrent queue is used to allow the output from the first thread to be used as input into the second. Describe, with justification, a situation where increasing the size of the buffer in the concurrent queue would improve overall system throughput. [2 marks]

6 Security

(*a*) During a security review, you encounter the following C function, which may be called by untrusted code:

```
int table[800];

int insert_in_table(int val, int pos) {
    if (pos > sizeof(table) / sizeof(int)) return -1;
    table[pos] = val;
    return 0;
}
```

Identify potential vulnerabilities and provide a fixed version. [4 marks]

(*b*) When you log into a system that uses Kerberos authentication by providing a password (e.g., using the Unix/Linux/macOS command `kinit`):

(*i*) What data does your computer send to and receive from the key-distribution centre (KDC) and which parts of that data are encrypted with whose key or password? [9 marks]

(*ii*) What is the purpose of a ticket-granting ticket? [2 marks]

(*iii*) Where are Kerberos tickets stored? [2 marks]

(*c*) Suggest and briefly explain three countermeasures used by a typical Linux distribution to mitigate the risk of stack-overflow vulnerabilities in included software. [3 marks]

7 Security

(*a*) (*i*) What effect does the Unix/Linux/macOS system call `chroot` have (or the GNU/Linux command-line tool of the same name)? [2 marks]

(*ii*) What kinds of resource can chroot restrict access to? How can the developer of a program P use `chroot`? How can the user of a program P use `chroot`? [4 marks]

(*iii*) Why would a developer or user of a program want to do this? Give a concrete example. [4 marks]

(*iv*) Name two other kinds of resource on a Unix system for which access is not affected by `chroot`. [2 marks]

(*b*) User `jane` types the following three commands into her Linux shell:

```
$ id
uid=1002(jane) gid=1002(jane) groups=20(dialout),513(staff)
$ ls -l ptool
-rwsr-xr-x 1 ptusr ptgrp 59640 Mar 22  2020 ptool
$ ./ptool
```

(*i*) State the various user and group identities associated with the started `ptool` process, by copying and completing the following table:

	real	effective	saved
user ID			
group ID			
supplementary groups			

[4 marks]

(*ii*) Which values is the `ptool` process permitted to provide in the `seteuid()` system call? [2 marks]

(*c*) Microsoft's Active Directory Domain Service stores information about users and computers in an LDAP object tree. It controls access to such objects using an extension of the access-control list mechanism also used for Windows NTFS files. What additional field does Active Directory ACEs use compared to NTFS ACEs and what is its purpose? [2 marks]

SECTION B

8 Semantics of Programming Languages

(a) Suppose we have a language with booleans, integers, and *mutable* variables:

$$\begin{array}{rcl} e & ::= & n \mid e_0 + e_1 \mid e_0 < e_1 \mid \mathsf{true} \mid \mathsf{false} \mid \mathsf{if}\ e\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 \\ & \mid & x \mid \mathsf{var}\ x = e_0\ \mathsf{in}\ e_1 \mid x := e \end{array}$$

(i) Give a grammar for the *values* of this language. [1 mark]

(ii) What mathematical object should be used to represent a store σ (which tracks which values each variable has)? [1 mark]

(iii) Give a reasonable operational semantics for this language, as a transition relation. (You may assume the existence of a substitution operation $\{v/x\}e$.)

$$\langle \sigma; e \rangle \rightsquigarrow \langle \sigma'; e' \rangle$$

This semantics should ensure (though you need not prove) that for any configuration $\langle \sigma; e \rangle$, it is either of the form $\langle \sigma; v \rangle$ with no further transitions, or otherwise it has at most one transition $\langle \sigma; e \rangle \rightsquigarrow \langle \sigma'; e' \rangle$. In addition to the formal rules, give an explanation of the reduction rules you define for variable declarations $\mathsf{var}\ x = e_0\ \mathsf{in}\ e_1$ and assignments $x := e$.

[8 marks]

(b) (i) Define a reasonable set of types for this programming language. [1 mark]

(ii) Explain what a typing context should look like for this language. [1 mark]

(iii) Define a set of typing rules for this programming language, which should ensure type safety. [7 marks]

(iv) State (but do not prove) the progress and type preservation theorems for this language. [1 mark]

9 Semantics of Programming Languages

Many languages (like C and Java) support *coercions*, in which values of one datatype (e.g., machine integers) can be used where values of another datatype (e.g., floating point numbers) are expected, by having the compiler silently insert code to convert from one type to another. Suppose we have a language with the following grammar of types:

$$T ::= \mathsf{int} \mid \mathsf{bool} \mid \mathsf{string} \mid T \times T' \mid T \to T'$$

Suppose we then define a subtyping relation as follows:

$$\frac{}{T \leq T} \qquad \frac{T \leq T' \qquad T' \leq T''}{T \leq T''}$$

$$\frac{T_1 \leq T_1' \qquad T_2 \leq T_2'}{T_1 \times T_2 \leq T_1' \times T_2'} \qquad \frac{T_1' \leq T_1 \qquad T_2 \leq T_2'}{T_1 \to T_2 \leq T_1' \to T_2'}$$

$$\frac{}{\mathsf{bool} \leq \mathsf{string}} \qquad \frac{}{\mathsf{int} \leq \mathsf{string}}$$

$$\frac{}{\mathsf{bool} \leq \mathsf{int}}$$

(*a*) Assuming the existence of functions `bool_to_string`, `int_to_string`, and `bool_to_int`, adapt the relation above to define a new relation $T \leq T' \rightsquigarrow e$, where e is a *coercion*, a closed function of type $T \to T'$. (You may use ML or lambda-calculus notation to define the coercions e.) [10 marks]

(*b*) Explain what this relation could be used for in a language implementation. [2 marks]

(*c*) Give definitions of `bool_to_string` and `bool_to_int`, and then use the relation you defined to give two subtyping derivations $\mathsf{bool} \leq \mathsf{string} \rightsquigarrow e_1$ and $\mathsf{bool} \leq \mathsf{string} \rightsquigarrow e_2$ such that e_1 and e_2 have different behaviour. [5 marks]

(*d*) What problem would this lead to in a language implementation? [3 marks]

END OF PAPER