CST1
COMPUTER SCIENCE TRIPOS Part IB

Monday 8 June 2026 14:00 to 17:00

COMPUTER SCIENCE Paper 4

*Answer **five** questions.*

*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

1 Compiler Construction

The following ambiguous grammar G_1 describes nameless lambda calculus terms. (Familiarity with nameless lambda calculus is not assumed.)

$\texttt{E} \to \texttt{A}$ $\quad$ $\texttt{E} \to \lambda\ \texttt{E}$ $\quad$ $\texttt{E} \to \texttt{E}\ \texttt{A}$ $\quad$ $\texttt{A} \to n$ $\quad$ $\texttt{A} \to \texttt{(}\ \texttt{E}\ \texttt{)}$

E is the start symbol, λ, n, (and) are terminals, and n stands for natural numbers (0, 1, 2, ...).

(*a*) Give an example of an *unambiguous* term that uses all the productions of G_1.

[2 marks]

(*b*) Give an example of an *ambiguous* term that uses all the productions of G_1.

[2 marks]

(*c*) Give two distinct leftmost derivations for your ambiguous term from Part (*b*).

[4 marks]

The ambiguity in G_1 is removed to give the following unambiguous grammar G_2:

$\texttt{E} \to \texttt{F}$ $\quad$ $\texttt{E} \to \lambda\ \texttt{E}$ $\quad$ $\texttt{F} \to \texttt{A}$ $\quad$ $\texttt{F} \to \texttt{F}\ \texttt{A}$ $\quad$ $\texttt{A} \to n$ $\quad$ $\texttt{A} \to \texttt{(}\ \texttt{E}\ \texttt{)}$

(*d*) Show that G_2 is SLR(1). [6 marks]

(*e*) Show that G_2 is not LL(1). [2 marks]

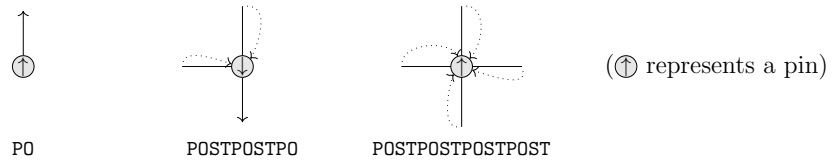
(*f*) Give an LL(1) grammar for the language accepted by G_2, and sketch a proof that your grammar is LL(1). [4 marks]

2 Compiler Construction

PINPEN programs control a pen and a set of pins on a plane. The pin and pens each have a position and direction. Programs are sequences of the following commands:

`P`	Places a new pin with the pen's current position and direction.
`O`	Moves the pen one step. Also draws a line from the old position to the new position.
`S`	Sets the pen's position and direction to the last-placed pin. Removes the pin. Does not draw anything.
`T`	Turns the pen 90° anti-clockwise.

For example, the program `POSTPOSTPOSTPOST` draws a `+` symbol as shown below:



(*a*) Given the following API:

```
type pos = int * int            val step : dir -> pos -> pos
type dir                        val draw : pos -> pos -> unit
type command = P | O | S | T    val turn : dir -> dir
```

write a PINPEN interpreter with type `command list -> pos * dir -> unit` using exceptions to implement `P` and `S`, and without using references. The second argument represents the pen's position and direction. [6 marks]

(*b*) Convert your PINPEN interpreter to use a continuation argument rather than exceptions to implement `P` and `S`, giving the interpreter the following type:

```
command list -> pos * dir -> (command list -> unit) -> unit
```

[6 marks]

(*c*) State whether each of the following PINPEN optimisations is valid:

(i) `PPOSS` $\rightsquigarrow$ `POS` (ii) `POS` $\rightsquigarrow$ `O` [2 marks]

(*d*) Describe *all* the sequences of commands that can be removed from *any* PINPEN program without changing the program's behaviour. [6 marks]

3 Semantics of Programming Languages

Many languages (C, C++, Rust, LLVM IR, etc.) deem some programs to have undefined behaviour (UB). This is done to permit compiler optimisations that are sound only under the assumption that the source program has no UB.

(*a*) Give a semantics for the following C-ish language in which it is UB to use a pointer outside its original allocation. Track the original allocations of pointers in the semantics with pointer values $p ::= (id, n)$ that pair an allocation ID id : ID with an address $n : \mathbb{N}$. The semantics should define a small-step transition relation $A, M, e \longrightarrow A', M', e'$ and an UB relation $A, M, e \longrightarrow \mathbf{UB}$, where A : ID $\rightharpoonup \mathbb{B} \times \mathbb{N} \times \mathbb{N}$ is an *allocation environment* that for each allocation created so far gives its metadata (*live*, *addr*, *size*), and memory $M : \mathbb{N} \to \mathbb{N}$ holds a natural number at each natural-number address. Use evaluation contexts. A source program is *closed* if it does not contain any free variables or literal pointer values. The semantics should either reduce or flag UB for any non-value closed e.

$$
\begin{array}{lll}
e ::= & v \mid e + n \mid *e \mid *e = e' \mid \mathbf{malloc}\,(e) \mid \mathbf{free}\,(e) \mid x \mid \mathbf{let}\, x = e\, \mathbf{in}\, e' \\
v ::= & n \mid p
\end{array}
$$

[14 marks]

(*b*) Explain briefly how your semantics handles use-after-free and use-after-reallocation, with examples of closed programs that exhibit UB because of each.

[3 marks]

(*c*) The top-level semantics of a closed program e are defined to be UB if any execution flags UB, and otherwise the set of integers it can evaluate to:

$$
\begin{array}{ll}
\mathbf{UB} & \text{if } A_0, M_0, e \longrightarrow^* \longrightarrow \mathbf{UB} \\
\{n' \mid \exists A', M'. A_0, M_0, e \longrightarrow^* A', M', n'\} & \text{otherwise}
\end{array}
$$

where $A_0 = \{\}$ and M_0 maps all addresses to 0.

Discuss when it is sound to hoist a load, for example optimising

$$\mathbf{let}\, x = e\, \mathbf{in}\, \mathbf{let}\, y = *z\, \mathbf{in}\, e'$$

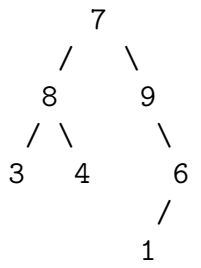
to

$$\mathbf{let}\, y = *z\, \mathbf{in}\, \mathbf{let}\, x = e\, \mathbf{in}\, e'$$

[3 marks]

4 Prolog

Consider a binary tree whose nodes are either atom `lf` for a leaf or compound term `n(Value,Left,Right)` where `Value` is an integer and `Left` and `Right` are trees. For example `n(7,n(8,n(3,lf,lf),n(4,lf,lf)),n(9,lf,n(6,n(1,lf,lf),lf)))`. represents the tree



In your answers ensure each relation has a comment giving a declarative reading of its behaviour. Avoid unnecessary use of *cut* or other extra-logical relations. The library relations `\=` and `is` may be used. Other library relations should not be assumed.

(*a*) Write a relation `sum(T,Sum)` which, when given an input tree `T`, will succeed with `Sum` set to the integer sum of all values in the tree. [2 marks]

(*b*) Write a relation `summed_tree(Tree,SummedTree)` which will succeed with `SummedTree` having the same structure as `Tree`, but with each value being the sum of the sub-tree at the corresponding position in the input `Tree`. [3 marks]

(*c*) Write a relation `breadth_first(T,V)` which, given an input tree `T` and variable `V`, will succeed on each redo with `V` being each value in the tree found in breadth-first left-right order. In the example tree, this means visiting the nodes in the following order: 7,8,9,3,4,6,1. [4 marks]

(*d*) Write a relation `root_path(T,V,Path)` which, given a tree `T` and a value `V` unique within that tree, will succeed with `Path` being the list of values from the root of `T` to the node containing value `V`, inclusive of the end-points. For example, if `T` is set to the example tree then `root_path(T,4,Path)` succeeds with `Path = [7,8,4]`. [4 marks]

(*e*) Assume a tree including unique values `V1` and `V2` from which root paths `P1` and `P2` respectively have been derived as in Part (*d*). Write a relation `route(P1,P2,Route)` which will succeed with `Route` set to the shortest list of values from `V1` to `V2` including those end-points. For example, `route([7,9,6],[7,8,4],Route)` succeeds with `Route = [6,9,7,8,4]`. [7 marks]

5 Programming in C and C++

A data structure for a LISP-like interpreter is specified in OCaml as follows:

```
type lisp_t =
  | Cons of lisp_t * lisp_t   | SAtom of string
  | Nil                       | IAtom of int
  | Ref of lisp_t ref (* a mutable pointer *)
```

(*a*) Without using C++, implement `lisp_t` in the C language, using one (or more) `struct`s or `union`s. If an `enum` is used to distinguish the variant forms, does it need a `Nil` member? [5 marks]

(*b*) The LISP language would store the list `[1,2]` as `Cons(IAtom 1, Cons(IAtom 2, Nil))`. However, in C, three further representations of lists are arrays of values, arrays of references and the intrusive representation (pointer in the datum/record). Briefly distinguish these three further representations, discussing any advantages for memory or runtime efficiency and whether an item can be in more than one list. [6 marks]

(*c*) General-purpose garbage collectors (like Boehm) work for many simple C coding styles. They operate without knowledge of any user-defined `structs` such as `struct lisp_t`, but they may require heap storage to be allocated with their own variant of `malloc`. Are there likely to be problems for your `lisp_t`? Is the presence of any `enum`s, `union`s, or mutable cells a help or hindrance? [5 marks]

(*d*) If your `struct lisp_t` structure were instead implemented in C++, what changes would be desirable? For C++, is distinguishing variants using an `enum` a good approach? [4 marks]

6 Programming in C and C++

Note: Part (*a*) of this question should be answered **without any reference to C++**.

(*a*) The C language has hardly any native support for character strings, with facilities mainly being provided by standard libraries.

(*i*) What language features or behaviours *must* a C compiler implement with respect to strings? [2 marks]

(*ii*) When might it be helpful or harmful to make sure every string is held at most once in memory? [4 marks]

(*iii*) In a programming exercise you are asked to write a C routine that removes the filename suffix, i.e. the last dot and any chars after it. It must be implemented entirely as native C code, not using `malloc` or any libraries, for use cases such as

```
const char *sans_suffix = remove_suffix("foobar.txt");
```

Is this possible? Give justification for your answer, discussing the possibilities for using the stack or other dynamic allocation or static storage. Code is not necessary but could be included to explain your answer. [4 marks]

(*b*) For this part of the question, consider string processing in C++.

(*i*) Does C++ provide any additional native support for string processing with respect to C? [1 mark]

(*ii*) Assuming `std::string` is not available, a C++ class is needed that provides convenient RAII memory management for strings. This will automatically deallocate string memory (without relying on a garbage collector). Discuss whether it is possible in this scheme to ensure that each string is held only once. [3 marks]

(*iii*) Using `new char [n]` and `delete`, but without using any library functions, sketch C++ code that implements Part (*b*)(*ii*), with strings potentially stored more than once. Include an overload of the `+=` operator which extends a string with another one. Add comments or arrows or proof arguments that demonstrate that no deleted memory is read. [6 marks]

7 Cybersecurity

A Linux-based server hosts containers from various users. Each container has its own IP address. Users in group `webdev` may define host-level port redirects by adding a line with container IP, external port and internal port to the plain text file `/var/cfg/port/map.txt`. For example, the line "`12.34.56.78 80 8080`" requests that traffic for 12.34.56.78:80 be sent to 12.34.56.78:8080.

A root cron job regularly runs the bash script `/usr/local/bin/ptfwd` shown below, which creates the requested port mapping for each line of `/var/cfg/port/map.txt`. [*Note:* The bash builtin `read [variable ...]` reads a line from stdin, splits it into whitespace-delimited fields, and assigns the first field to the first variable, the second field to the second variable, and so on. If there are more fields than variables, the remaining fields and delimiters are assigned to the last variable. A backslash (\) removes any special meaning for the next character.]

```
while read IP EXTPORT INTPORT; do
    bash -c "/usr/sbin/iptables -t nat -A PREROUTING -d $IP -p tcp \
        --dport $EXTPORT -j DNAT --to-destination $IP:$INTPORT"
done  </var/cfg/port/map.txt
```

The following files and directories are on the system.

```
drwxr-xr-x 2 root  admin   4096 Aug 18  2020 /usr/local/bin
-rwxr-xr-x 1 root  root     209 Jan 15  2020 /usr/local/bin/ptfwd
drwxr-xr-x 2 root  admin  36864 Sep 23  2024 /usr/sbin
-rwxr-xr-x 1 root  root  224424 Apr  8  2024 /usr/sbin/iptables
drwxrwxr-x 2 bob   staff   4096 Apr 12  2016 /var/cfg
drwxr-xr-x 2 root  admin   4096 Apr 12  2016 /var/cfg/port
-rw-rw---- 1 root  webdev  3484 Dec  8  2025 /var/cfg/port/map.txt
```

(a) Give a complete description of a privilege escalation attack through which user `alice`, who is in group `staff` but not in `webdev`, `admin` or `root`, gains the ability to run arbitrary commands as root. First, clearly describe the overall strategy. Second, for each step of the attack, specify: (i) the exact Linux command the attacker executes, (ii) what the command accomplishes. [8 marks]

(b) Identify the critical vulnerabilities in the above configuration. Justify your answer by showing what step(s) of your attack in Part (a) would stop working if each vulnerability were eliminated, without impairing the desired functionality. For each identified vulnerability, describe how to remedy it. [8 marks]

(c) Consider now user `charlie`, who is in group `admin` but not in any of the others, and user `daria`, who is in `webdev` but not in any of the others. For each, if the attack is possible, repeat Part (a); else, convincingly argue that no such attack is possible. [4 marks]

8 Cybersecurity

As its CISO (Chief Information Security Officer), you are responsible for the security of Megacorp's commercially sensitive data. A compromise of such data would cost £5,000,000 in terms of regulatory fines, reputational damage and lost business. Your security audit highlighed three attack vectors and offered the following estimates.

(A) SQL Injection: Probability is 20% per year. Software mitigations, costing £55,000 initially and £15,000 per year thereafter for maintenance, would reduce this probability to 1% per year.

(B) Phishing: Probability is 35% per year. Security awareness training, costing £20,000 per year, would reduce it to 5% per year.

(C) XSS: Probability is 15% per year. Software mitigations, costing £45,000 initially and £20,000 per year thereafter for maintenance, would reduce it to 2% per year.

A breach via any of the three highlighted vectors, which are assumed to be independent of each other, will result in the full £5,000,000 loss.

[*Note:* The CEO might sometimes be wrong. In such cases, the job of a competent CISO is to educate and advise, on the basis of convincing reasoning. Exact numerical answers are required for full marks.]

(*a*) Compute the expected cost to the business in the first year if all mitigations are deployed. [4 marks]

(*b*) The CEO of Megacorp dislikes upfront expenses and is only inclined to implement at most two of the mitigations—preferably the cheaper ones. Compute the expected first-year cost to the business for each of the three pairs and prepare a well-justified recommendation for the CEO. [8 marks]

(*c*) Based on Prospect Theory, you anticipate that the CEO might have objections to your recommendation in Part (*b*). List any objections you anticipate, based on Prospect Theory. Explain, using Prospect Theory, why the CEO might raise them. Explain how you will address the objections to convince the CEO of the superiority of your recommendation. [4 marks]

(*d*) An insurance company is offering to cover the full £5,000,000 loss for an annual premium of £180,000. Explain to the CEO how this compares to the best of the previously considered solutions, both in the first year and in subsequent years, and recommend whether to accept or reject this offer. [4 marks]

END OF PAPER