COMPUTER SCIENCE TRIPOS Part IB

Monday 4 June 2012 1.30 to 4.30

COMPUTER SCIENCE Paper 3

*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 Algorithms II

(*a*) Draw a proto-vEB node, labelling each of its fields and briefly explaining what each field does. [2 marks]

(*b*) Draw a vEB node, labelling each of its fields and briefly explaining what each field does. [2 marks]

(*c*) On its own page, draw in pencil a complete and legible vEB tree holding the values $\{0, 2, 4, 8, 9, 10, 13, 14\}$. The correctness of the structure and the accuracy of all fields of all nodes is important. Once done, write each of the values in ink under the leaf record in which it is logically stored. [6 marks]

(*d*) Consider the task of inserting a value v into a proto-vEB or vEB tree whose root node is r. Assume the value is in range and not already in the tree. Write two legible pieces of high-level pseudocode for `insertInProtoVEB(r, v)` and `insertInVEB(r, v)` respectively. Clarity (insert comments where appropriate) will count more than perfect low-level accuracy. Derive the computational complexity of your two procedures using the appropriate recurrence formulae (but solving the recurrences is not required). Explain what specific features of the vEB tree make it faster than the proto-vEB tree for this particular task. [10 marks]

2 Algorithms II

(*a*) Briefly explain the concurrency keywords "spawn", "sync" and "parallel for". [3 marks]

(*b*) Define what a "determinacy race" is and list the precise circumstances under which one may occur. [3 marks]

(*c*) In less than 10 lines of pseudocode, give a clear example of a determinacy race (but *not* the same example as in the textbook or the course handout) and briefly explain why it is one. [3 marks]

(*d*) Do any of the two following pseudocode functions contain determinacy races? Justify and explain your answer for each of the two pieces of code.

```
def fibonacci(n):
    if n <= 1:
        return n
    else:
        x = spawn fibonacci(n - 1)
        y = fibonacci(n - 2)
        sync
        return x + y
```

```
def matrixMultiply(A, B):
    # PRECONDITION: A and B are square matrices of the same size
    n = A.rows
    let C be a new n x n matrix
    parallel for i = 1 to n:
        parallel for j = 1 to n:
            C[i,j] = 0
            for k = 1 to n:
                C[i,j] = C[i,j] + A[i,k] * B[k,j]
    return C
```

[8 marks]

(*e*) Give an example of a circumstance in which a multithreaded algorithm containing a determinacy race might still be correct. For each of the races you found and discussed in (*d*), if any, state whether this circumstance applies. [3 marks]

3 Programming in C and C++

In this question, where appropriate, you may use a short fragment of code to complement your explanation.

(a) (i) What is the difference between a local and global variable in C? (Consider variable scope, storage and initialisation.)

(ii) What are the properties of a static member variable in a C++ class?

[4 marks]

(b) (i) Briefly explain pointer arithmetic in C. Give an example code snippet involving pointers in which it would be *inappropriate* to use pointer arithmetic, and explain why.

(ii) Explain how in some respects pointers are equivalent to arrays, and give one respect in which they differ.

[4 marks]

(c) Explain why a function might be declared virtual in a C++ superclass.

[4 marks]

(d) (i) How does use of the void $*$ pointer in C allow a form of polymorphism? Give an example function declaration using the void $*$ pointer.

(ii) What is the main problem with the use of void $*$, and how does C++ improve on this? Give the improved function declaration in C++ for your example function in part (d)(i).

[4 marks]

(e) (i) Why might it be useful to define a copy constructor for a C++ class? Give an example of a copy constructor for a simple class.

(ii) Why might it be useful to explicitly define the assignment operator (=) for a C++ class? Give an example definition of the assignment operator for a simple class.

[4 marks]

4 Compiler Construction

(*a*) Define the following terms used when discussing a grammar:

(*i*) a non-terminal symbol [1 mark]

(*ii*) an ambiguous grammar [1 mark]

(*iii*) a production [1 mark]

(*iv*) a context-free grammar [2 marks]

(*v*) a regular grammar [2 marks]

(*b*) The following grammar defines a language where expressions are strings or integers. A type error arises when an integer is added to a string.

```
Var -> x | y
Exp -> Var | 0 | 1  | "cat" | "dog" | Exp + Exp
S   -> Var := Exp | S S
```

(*i*) Give a syntactically-correct sentence of the language that contains a type error. [1 mark]

(*ii*) What phase (or pass) of a typical, simple compiler would detect such a type error? Sketch the fragment of code that actually spots the error. [3 marks]

(*iii*) Provide a modified grammar such that type errors are also syntax errors. [3 marks]

(*iv*) Why is such a modified grammar generally impractical? [1 mark]

(*c*) Certain operators, such as **logical and**, which is commonly denoted with **&&** use short-circuit evaluation.

(*i*) Define short-circuit evaluation. [2 marks]

(*ii*) Give two reasons why it is useful. [1 mark]

(*iii*) Describe the problem, and its solution, that arises when a short-circuit operator is encountered during a simplistic compilation of a syntax tree to stack machine code. [2 marks]

5 Compiler Construction

(*a*) Define the following forms of variable: [1 mark each]

(*i*) statically allocated global variable

(*ii*) local variable (to a function)

(*iii*) free variable

(*b*) In an object-oriented language, which of the above three terms best describes a field? [1 mark]

(*c*) How are storage addresses typically allocated for each of these types of variable? Describe the stage/pass of the compiler that makes the allocation and also describe any changes made by the operating system linker or loader. [3 marks]

(*d*) What addressing mode or sequence of instructions is typically used in compiled code to read (or write) each of the variable forms from part (*a*)? [1, 1, 4 marks]

(*e*) With dynamic storage allocation, how does the memory manager 'new' operator know how much memory to allocate? Discuss both compilation to machine code and execution on a VM. [2 marks]

(*f*) Why is garbage collection easier to implement in strongly-typed languages (compared with weakly-typed languages)? [1 mark]

(*g*) What overheads does allocating arrays and class objects on the stack cause compared with placing them on the heap? What are the advantages and disadvantages of each method? [4 marks]

6 Concepts in Programming Languages

(*a*) Give an overview of the execution models (abstract machines) associated with Fortran, Lisp, Algol-60, Pascal, C, ML and Java. (These are not necessarily distinct.) Mention how storage allocation and deallocation is performed. [4 marks]

(*b*) Explain to what extent the above languages:

(*i*) provide static scoping [2 marks]

(*ii*) provide static type checking [2 marks]

(*iii*) are type-safe—for each language either state it to be type-safe or sketch a type-unsafe program [4 marks]

(*c*) In Algol, functions can be passed as arguments to functions but the type only mentions the result type of such a function formal parameter, whereas in ML more detail is given. Which is better, and why? [2 marks]

(*d*) Array types in object-oriented languages can be seen as generic classes having read and write operations. Java arrays are described as *covariant*. Contrast this with *non-variant* arrays in terms of compile-time versus run-time type errors. [2 marks]

(*e*) For each of the following ML declarations, either justify their ability to be soundly used or give a program using them which would violate type safety:

(*i*) `exception poly of 'a;` [1 mark]

(*ii*) `val ml = ref [];` [1 mark]

(*f*) Explain why the '`private`' keyword is commonly available in object-oriented languages. Giving reasons, explain whether it is possible for the value of a private field of one object initialised by a parameterless constructor to be accessed from another object of the same class. [2 marks]

7 Further Java

Consider the following interface definition representing a stack data structure:

```
public interface Stack<T> {
 public void push(T item);
 public T pop() throws java.util.NoSuchElementException;
}
```

(*a*) Provide a *non*-thread-safe implementation of the interface. Apart from `NoSuchElementException` you may not use classes from the Standard Library. [8 marks]

(*b*) Provide and explain an example execution trace which demonstrates that your implementation is not thread safe. [6 marks]

(*c*) Define fine- and coarse-grained locking for thread safety. [2 marks]

(*d*) Describe how you would change your implementation to use a coarse-grained locking strategy. Explain why your example execution trace can no longer occur. [2 marks]

(*e*) Would there be any significant performance benefit from using a fine-grained locking strategy? Explain why. [2 marks]

8 Prolog

(*a*) Give the result and any variable bindings that occur from making each of the following (independent) queries.

(*i*) `32 = A.` [1 mark]

(*ii*) `a(b(6, A)) = a(b(B, 2)).` [1 mark]

(*iii*) `a(b(x, A)) = a(b(5, 4)).` [1 mark]

(*iv*) `A = 3 + 6, A is 9.` [1 mark]

(*v*) `A is 3 + 6, A = 9.` [1 mark]

(*b*) Given the following clauses,

```
a(4).
a(x).
b(3,x).
b(1,7).
c(A, B, C) :- a(A), b(B, _), !, a(D).
c(A, _, B) :- b(A, B).
```

(*i*) List the solutions reported by Prolog to the query `c(P, Q, R)`, for each giving any binding of variables that occurs. [2 marks]

(*ii*) Explain whether the query in part (*b*)(*i*) can bind `P` to `x`. [1 mark]

(*iii*) List the solutions to the query `c(1, 6, R)`, for each giving any binding of variables that occurs. [1 mark]

(*c*) We represent a binary tree using a term of the form `[Left,NodeName,Right]` or `null`. Here is an example perhaps recording a tree of excursions between countries:

```
[[null,de,null],uk,[[null,ua,null],pt,null]]
```

Write a predicate `p(+Tree, +NNames, -Ps)` where `Tree` is a tree as above, `NNames` is a list of node names, and `Ps` is to be unified with a list of elements of the form `[N,P]` where `N` is in `NNames` and `P` is `N`'s parent in `Tree`. [For example, supposing `T` is bound to the above tree, the query `p(T, [ua, za], Ps)` binds `Ps` to `[[ua, pt]]`.] [5 marks]

(*d*) Difference lists are a powerful tool for increasing efficiency but they address a very specific problem. Does this issue arise in your implementation of `p/3`? In other words, can `p/3` be made more efficient using difference lists and why (or why not)? [2 marks]

(*e*) Regardless of your answer above, write a predicate `pdl/3` that behaves just like `p/3` but uses difference lists in its implementation. [4 marks]

9 Software Engineering

(*a*) Describe the following two methodologies as they are used in building safety-critical systems:

(*i*) *failure modes and effects analysis*;

(*ii*) *fault tree analysis.*

[3 marks each]

(*b*) Describe three cognitive factors that can lead to operator errors in safety-critical systems. [2 marks each]

(*c*) If the safety of a system depends on correct operator actions, then under what circumstances is fault tree analysis likely to be more or less effective?

[8 marks]

END OF PAPER