



M263/J



Course Examination 2004
Building blocks of software

Friday 15th October 2004 10.00 am – 1.00 pm

Time allowed: 3 hours

There are **TWO** parts to this examination. You should attempt **both** of them.

You should attempt **ALL** the questions in Part I.
You should attempt **TWO** questions from Part II.

In Part 1, the marks available for each question are shown alongside the question.
In Part 2, each question is worth 20 marks.

Part I carries 60% of the total marks and Part II carries 40%.
You are advised to spend about **100 minutes on Part I** and **70 minutes on Part II**, and to leave yourself about 10 minutes for checking.

If a question says “evaluate”, then you should show your working. If a question says “write down”, then full credit will be awarded for a correct answer.

At the end of the examination

Check that you have written your personal identifier and examination number on each answer book used. **Failure to do so will mean that your work cannot be identified.** Put all your used answer books and your question paper together with your signed desk record on top. Fix them all together with the paper fastener provided.

Write the numbers of the questions that you have attempted in the spaces provided on the front of the answer book.

Part I

You should attempt **ALL** the questions in this part and are advised to spend about **100 minutes** on it. This part carries 60% of the marks for the whole examination.

The marks for each question are given beside the question.

Question 1 (4 marks)

Let *aStack* be the stack of integers [1,2,3,4]. Write down the value of each of the following expressions.

- (a) *PEEK*(*aStack*).
- (b) *POP*(*aStack*).
- (c) *PEEK*(*POP*(*aStack*)) == *PEEK*(*aStack*).

Question 2 (6 marks)

The code below implements a function *QU2* which has as input an array of integers, and which returns an integer.

- (a) Trace the values of the variables *i* and *anInt* after each execution of statement (1) as this code is executed when the input array *arr* is [1,5,10,3,8]. Write down the value that is returned for this input.
- (b) Give a precondition that the input *arr* must satisfy, in order that the code in the given implementation is valid.

```
function QU2(arr)
{
  var i in Int
  var anInt in Int
  anInt <-- 0
  for (i <-- 1 to 3)
  {
    anInt <-- anInt + AT(i,arr)    //(1)
  }
  return anInt
}
```

Question 3 (5 marks)

The function *QU3* has a full array of integers as input, and returns **true** if the first and last numbers in the array are the same and **false** if they are different.

- (a) Write down a suitable specification for *QU3*. (Your specification should give: a signature line; a precondition, and a postcondition.)
- (b) Write down an implementation for *QU3*.

Question 4 (4 marks)

A function QU_4 is specified below, and is followed by a fragment of code using this function. Write down the states of the variable s after execution of each of statements (1) and (2) in the fragment.

function $QU_4(n$ in **Int**, s in **Spot**) **return** in **Spot**

pre $n \geq 0$.

post The returned value is an object of type **Spot** that is n places to the right of s and 2 places below it.

```
var s in Spot
s <-- QU4(4,s)           //(1)
s <-- QU4(10,s)         //(2)
```

Question 5 (3 marks)

Suppose that t is a variable of type **Turtle**. Write down a fragment of code that is marker-state invariant, and adds to the diagram of t a line of five spots starting at the current position of the marker and going in the current marker direction.

Question 6 (4 marks)

Trace the states of the variables v and s as the code fragment below is executed. Your trace should show the states after each execution of statement (1), and after execution of statement (2).

```
var i in Int
var s in mString
var v in VectorPlus of mString
s.setString("a")
for (i <-- 1 to 2)
{
  v.addFirst(s)
  s <-- s.charAppend('q')   //(1)
}
v.sort()                   //(2)
```

Question 7 (4 marks)

Suppose that A and B are the sets of integers enumerated below.

$$A = \{11, 14, 15, 21\}$$

$$B = \{21, 22, 14, 23\}$$

(a) Write down each of the following sets by enumeration.

(i) $A \cup B$

(ii) $A \cap B$

(iii) $A - B$

(b) Write down the cardinality of $A \cup B$.

Question 8 (5 marks)

Suppose that t is a variable of type **BinTree of Int** whose state is as shown in Figure 1. For each of the method calls in (a)–(c), either write down the value that will be returned, or explain why the call is not valid.

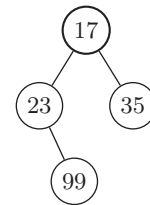


Figure 1

- (a) $t.leftTree()$
- (b) $t.rightTree().root()$
- (c) $t.isEmpty().leftTree()$

Question 9 (4 marks)

Give a truth table for the Boolean expression $b \wedge \neg(a \wedge b)$, and hence write down the disjunctive normal form of this expression.

Question 10 (5 marks)

The set *Form* is the set of pupils in a particular school form (identified as strings). Certain predicates concerning pupils x and y are defined below.

- $GIRL(x)$: x is a girl.
- $TALLER(x, y)$: x is taller than y .
- $UNDER(x)$: x is less than 9 years old.

Express each of the following propositions in natural language.

- (a) $GIRL(\text{"Sam"}) \wedge UNDER(\text{"Sam"})$
- (b) $\exists x \text{ in } Form[\neg GIRL(x) \wedge UNDER(x)]$
- (c) $\forall x \text{ in } Form[TALLER(\text{"Sam"}, x) \Rightarrow GIRL(x)]$

Question 11 (3 marks)

A relation scheme **CUSTOMER** is described below. In this:

- $custID$ is a unique identifier of a customer;
- $custAdd$ gives the customer's address;
- $contact$ gives the staff ID code of the member of staff who is the customer's primary contact;
- $staffExt$ gives the telephone extension of a member of staff.

Write down the explicit functional dependencies that follow from the rules given for **CUSTOMER**.

CUSTOMER		Relation Scheme
Attribute	Domain	Type
$custID$	$CustomerID$	Int
$custAdd$	$Address$	mString
$contact$	$StaffID$	Int
$staffExt$	$Extension$	Int
<i>Rules</i>		
R1 A customer has a single address.		
R2 A member of staff has a single telephone extension.		
R3 A customer has a single primary contact.		

Question 12 (5 marks)

A recursive implementation *recQU12* is given below for a function *QU12* with input *v* of type **Vector of Int** and return value of type **Int**. Use *recQU12* to evaluate *QU12(v)* when *v* is the vector $[-10, -5, 8, 2]$. That is, determine the value returned for this input vector *v*. Each step in your evaluation should involve a single use of one of statements (1), (2) or (3), and you should say which of these statements is used at each step.

```
function QU12(v)
{
    // recQU12
    var temp in Vector of Int
    temp <-- v.clone()
    if (temp.size()== 0) then
    {
        return 1 // (1)
    }
    else
    {
        if (temp.at(1) > 0) then
        {
            return 1 // (2)
        }
        else
        {
            temp.removeFirst()
            return 1 + QU12(temp) // (3)
        }
    }
}
```

Question 13 (5 marks)

A function *COUNTPOS* is specified below, followed by an implementation *impCountPos*. (For this question, interpret “the integer n is positive” as meaning “ $n > 0$ ”.)

function *COUNTPOS*(s in Stack of Int) **return** in Int

pre true.

post The returned value is the number of positive integers that appear in s . (If there are no positive integers in s , or if s is empty, then the returned value is 0.)

```
function COUNTPOS(s)
{
    //impCountPos
    var count in Int
    var toDo in Stack of Int
    count <-- 0
    toDo <-- s
    while (NOT(SIZE(toDo) == 0))
    {
        if (PEEK(toDo) > 0) then
        {
            count <-- count + 1
        }
        toDo <-- POP(toDo)
    }
    return count
}
```

(a) In the implementation, show that the condition

$$\text{count} + \text{COUNTPOS}(\text{toDo}) = \text{COUNTPOS}(s) \quad (\text{LIC})$$

holds just before the **while** loop is entered. (Remember, in this condition *COUNTPOS* is a reference to the function as given in the specification.)

(b) You may assume that (LIC) is a loop invariant condition for the **while** loop in *impCountPos*. (You do **not** need to prove this.) Given that the loop invariant condition holds when the loop is left, complete a proof that the implementation *impCountPos* is correct.

Question 14 (3 marks)

The time complexity functions of two different implementations, A and B, of the same function are given below.

For implementation A: $f(n) = 2n \log_2(n) + 200n$

For implementation B: $g(n) = 25n \log_2(n) + 3n^2$

(a) Write down the order of each of f and g .

(b) Comment briefly on the significance of your classification in part (a).

[3 marks]

END OF PART A

Part II

You should attempt **TWO** questions from this part. You are advised to spend about **70 minutes** on Part II.

Each question carries 20% of the marks for the whole examination, and an indication of the allocation of marks within each question is given beside the question.

Question 15 (20 marks)

- (a) Let *Students* be the set of students in a particular school year, and let predicates concerning these students have the following interpretations.

TENNIS(*x*): *x* has played for the school tennis team;

MATHS(*x*): *x* has passed maths A level.

- (i) Show that the following equivalence holds. Use the rules of Boolean algebra as given in the *M263 Course Handbook*, and say which rule is used at each step.

$$\textit{TENNIS}(x) \Rightarrow \neg \textit{MATHS}(x) \equiv \neg(\textit{TENNIS}(x) \wedge \textit{MATHS}(x))$$

- (ii) Express formally the proposition *p* below. Give a formal expression that reflects directly the way the proposition is expressed in English. Use any of: the predicates *TENNIS* and *MATHS*, the set *Students*, and the connectives $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow, \forall, \exists$.

p: It is not true that, for all students in the set *Students*, if the student has played in the school tennis team then they have not passed maths A level.

Give an alternative, and simpler, expression of *p* in English. Give an alternative formal proposition that reflects this alternative form of English. Show, using equivalences given in the *M263 Course Handbook*, that your two formal propositions are equivalent. [7 marks]

- (b) Prove by induction that, after execution of the code fragment below, the state of *aSpot* is $(2n, n)$. Here, *n* is an integer with $n > 0$.

State clearly the proposition *p*(*j*) (where $j \geq 0$ is an integer) whose truth is established by your proof.

```
var aSpot in Spot
var i in Int
for (i <-- 1 to n)
{
    aSpot.upOne()
    aSpot.rightOne()
    aSpot.rightOne()
}
```

[9 marks]

- (c) An argument is given below, and some simple propositions that can be used to express the argument formally. Using the rules of inference given in the *M263 Course Handbook*, show that the argument is valid. (State which rule is used at each step.)

Argument

Both Jack and Jill will go up the hill. (1)

If Jack goes up the hill then he will fall. (2)

Either Jack will not fall or the water will be spilt. (3)

Hence the water will be spilt. (4)

Simple propositions

jack: Jack will go up the hill.

jill: Jill will go up the hill.

fall: Jack will fall.

spilt: The water will be spilt. [4 marks]

Question 16 (20 marks)

A function *ISIN* is specified below.

function *ISIN*(*s* in **mString**, *c* in **Char**) **return** in **Bool**

pre **true**.

post The returned value is **true** if the character *c* appears one or more times in the string *s*, and is **false** if *c* does not appear at all in *s*.

- (a) A partial implementation of *ISIN* is given below. Write down the code required at each of positions (1)-(4) in order to complete this implementation.

```
function ISIN(s,c)
{
var i in Int      // the loop index
var found in Bool // is c found yet?
(1)              // initialise found
for (2)          // work through s starting at the front
{
  if (3) then    // if the current character in s is c
  { (4) }        // then set found
}
return found
}
```

[7 marks]

- (b) Using logic notation, and methods of the relevant classes, express formally the following predicate about variables *s* of type **mString** and *c* of type **Char** (where *s* is not empty).

The character *c* appears one or more times in the string *s*.

[4 marks]

- (c) An alternative partial implementation of *ISIN*, using recursion, is given below. Write down the code required at each of positions (1)-(3) in order to complete this implementation.

```
function ISIN(s,c)
{
var b in Bool
if (1) then // if s is empty
{ (2) }     // set b to the stopping value
else
{ (3) }     // set b using a recursive call
return b
}
```

[5 marks]

- (d) A new class **QU16** extends the class **mString** by adding a single method, specified below.

method *charIn*(*c* in **Char**) **return** in **Bool**

pre **true**.

post The returned value is **true** if the character *c* appears one or more times in the receiving string *s*, and is **false** if *c* does not appear at all in the receiver.

Complete the implementation of the class **QU16** started below (given in a form that could be used in the WorkPad).

```
Class QU16
{
  this.charIn calls charInQU16
}
```

[4 marks]

Question 17 (20 marks)

A distance learning institution keeps information about end of year examinations in a relation over a relation scheme **EXAM**. The attributes of **EXAM** are

studID, *courseID*, *date*, *venue*, *contactID*, *contactPhone*.

A tuple over **EXAM** gives: the student identifier, the identifier of a course the student is taking, the date of the examination for that course, the examination venue, the identifier of a contact person for that venue, and the telephone number of the contact person. We assume that no two students have the same student identifier, that no two courses have the same course identifier, and that no two contact persons have the same contact identifier.

The rules of the scheme are given below, followed by part of a sample relation over the scheme.

Rules

1. Each course is examined on only one day.
2. Each student is assigned to a particular examination venue for the examination on a particular course (but the venue may be different for different courses).
3. Each venue has a single contact person.
4. Each contact person has a single telephone number.
5. Each telephone number is for a unique contact person.

	<i>studID</i>	<i>courseID</i>	<i>date</i>	<i>venue</i>	<i>contactID</i>	<i>contactPhone</i>
t1	Smith101	m263	15Oct	Nottingham	HoodR1	1515999000
t2	Smith101	t888	16Oct	Derby	AttilaTH2	11223344
t3	Jones22	t777	16Oct	Derby	AttilaTH2	11223344
t4	Jones22	t123	16Oct	Derby	AttilaTH2	11223344
t5	Patel14	m263	15Oct	London	KhanG1	99887766

- (a) By giving appropriate tuples from the table above, show that none of the following is a functional dependency of the scheme **EXAM**.
- (i) $\{studID\} \rightsquigarrow \{venue\}$.
 - (ii) $\{courseID\} \rightsquigarrow \{studID\}$.
 - (iii) $\{studID, date\} \rightsquigarrow \{courseID\}$. [3 marks]
- (b) Write down the explicit functional dependencies that follow from the given rules about the scheme. [4 marks]
- (c) Explain why $\{studID, courseID\}$ is a key of **EXAM**. [3 marks]
- (d) Write down an example of a transitive dependency of **EXAM**. [2 marks]
- (e) Explain why the scheme **EXAM** is not in second normal form, and show how to divide it into schemes that are in second normal form. (You may assume that $\{studID, courseID\}$ is the *only* key of **EXAM**.) [3 marks]
- (f) For the schemes you gave in your division in part (e), determine whether each is in third normal form. If they are not, show how to divide them into schemes that are in third normal form. [3 marks]
- (g) In part (f) you gave a division of **EXAM** into schemes that are in third normal form. Describe any associations between those schemes, giving all the foreign keys. [2 marks]

Question 18 (20 marks)

A specification and implementation of a function *TENINBST* are given below.

function *TENINBST*(*t* in **BinTree of Int**) **return** in **Bool**

pre *t* is a binary search tree.

post The returned value is **true** if *t* contains one or more copies of the integer 10, and is **false** otherwise.

```
function TENINBST(t)
{
    //recInBst
    if (t.getDepth() == 0) then                //(1)
    { return false }                          //(2)
    else
    {
        if (t.getRoot() == 10) then           //(3)
        { return true }                       //(4)
        else
        {
            if (t.getRoot() > 10) then        //(5)
            { return TENINBST(t.leftTree()) } //(6)
            else
            { return TENINBST(t.rightTree()) } //(7)
        }
    }
}
```

- (a) Suppose that *t* is the tree given in Figure 2. Trace the execution of *recInBst* with this input, showing which of statements (1)-(7) is executed at each step. What is the total number of statement executions in this case? [4 marks]

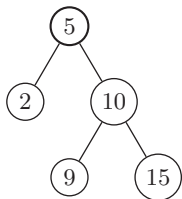


Figure 2

- (b) Let $T(n)$ be the total number of statement executions when *recInBst* is executed for a tree *t* of depth *n* in the worst case.

- (i) What is the “worst case” for this implementation?
(ii) Explain why $T(n)$ satisfies the recurrence system given below.

$$\begin{aligned} T(0) &= 2 \\ T(n) &= T(n-1) + 4 \quad (n > 0) \end{aligned}$$

- (iii) Find a solution for the recurrence system given in (ii). Classify $T(n)$ as $\Theta(f(n))$ for a suitable function *f* in the hierarchy given in the *M263 Course Handbook*.

- (iv) Compare the value given for $T(3)$ by your solution, and the count of statement executions in (a). Comment briefly on this comparison. [10 marks]

- (c) For an integer *n*, with $n \geq 0$, $F(n)$ is defined by the formula $F(n) = 5 \times 2^n - 3$, and $U(n)$ is defined by the recurrence system below.

$$\begin{aligned} U(0) &= 2 \\ U(n) &= 2 \times U(n-1) + 3 \quad (n > 0) \end{aligned}$$

- Prove by mathematical induction that $U(n) = F(n)$ (for all integers $n \geq 0$). [6 marks]

[END OF QUESTION PAPER]