E2.7 (A) + (B) together

E1.8
E2.7A

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2003

SOFTWARE ENGINEERING: INTRODUCTION, ALGORITHMS AND DATA STRUCTURES

Monday, 9 June 2:00 pm

Time allowed: 1:30 hours

There are THREE questions on this paper.

Answer TWO questions.

This exam is OPEN BOOK

Corrected Copy

Any special instructions for invigilators and information for candidates are on page 1.

Examiners responsible First Marker(s) : M.P. Shanahan
Second Marker(s) : Y.K. Demiris

Information for Invigilators:

Students may bring any written or printed aids into the exam.

Information for Candidates:

None.

The Questions

1. Assume the existence of a data type `TList` for a linked list, with the standard set of access procedures `Empty`, `First`, `Rest`, and `Add`. Now consider the following procedure.

```
function Merge(List1, List2 : TList): TList;
  var List3: TList;
  begin
    List3 := Empty;
    while List1 <> Empty or List2 <> Empty do
    begin
      if List1 = Empty or
        First(List2) < First(List1)
      then begin
        List3 := Add(First(List2),List3);
        List2 := Rest(List2);
      end
      else begin
        List3 := Add(First(List1),List3);
        List1 := Rest(List1);
      end;
    end;
    Merge := List3;
  end;
```

(a) Suppose `L1` is the list `[Amber, Chris, Ellen]` and `L2` is the list `[Billy, Darren]`. Trace the execution of the procedure call `Merge(L1,L2)` by showing the values of `List1`, `List2`, and `List3` at the end of each iteration of the while loop. [10]

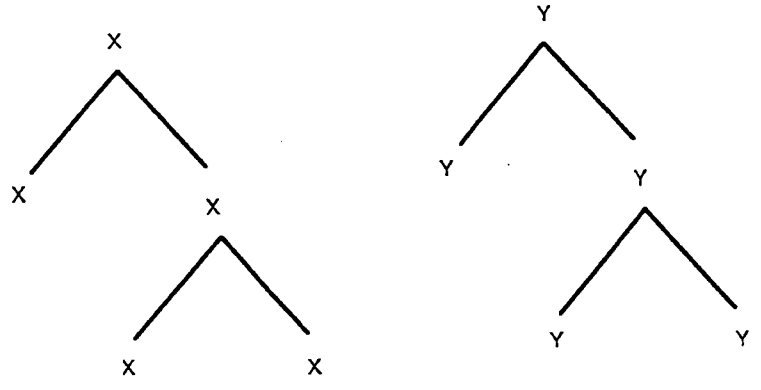
(b) What does the function do? How does it work? What conditions must `List1` and `List2` meet for the function to work correctly? [6]

(c) What difference, if any, would it make to the procedure if the arguments were declared as call-by-reference? [4]

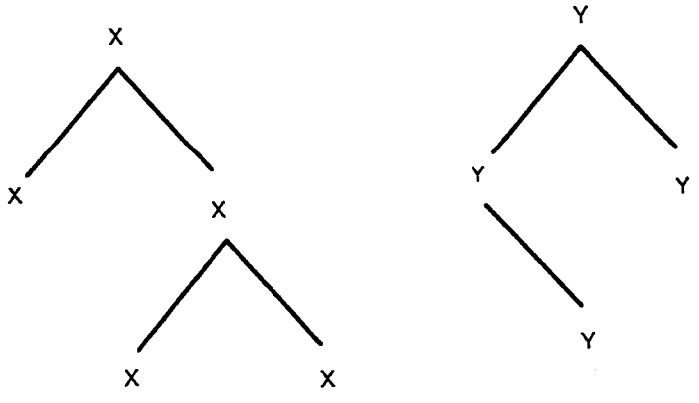
2. (a) Write a procedure that takes a two-dimensional array A1 of N by M integers and produces another N by M array A2 of integers in which each element [x,y] has been replaced by the average of the neighbourhood of five elements comprising [x,y] itself and the four elements above, below, and to either side of [x,y]. Copy the edges and corners of A2 straight from A1. [7]

(b) Assuming the compiler does not carry out any optimisation, how many add instructions *in total* will be executed by the procedure when it runs? Explain your workings. [7]

(c) Assume the existence of a data type `TTree` for a binary tree with the standard set of access procedures `EmptyTree`, `Left`, `Right`, and `Root`. Write a function `SameShape` that takes two binary trees and returns True if they have the same shape, ignoring the content of their nodes, and False otherwise. For example, the following two trees have the same shape,



while the following two trees do not.



[6]

3. (a) Write a non-recursive function `Match` that takes a string and returns False if the string contains non-matching round parentheses, but True otherwise. For example, if the string is "((P+Q)–R)+1" then the function should return True, but if the string is "Happy(Fred))" or "Happy(Fred" then the function should return False. You can assume the existence of a function `Length(S)` that returns the length of the string `S`.

Hint: an easy way to do this is to work along the string, maintaining a count of the level of nesting. [6]

(b) The following code defines a recursive function with the same purpose as the function in part (a).

```
1 function Match(S : string;
2   I, J : integer): boolean;
3 begin
4   if I > Length(S)
5   then begin
6     if J = 0
7     then return True
8     else return False;
9   end
10  else if J < 0
11  then return False
12  else begin
13    if S[I] = '('
14    then return Match(S,I+1,J+1)
15    else if S[I] = ')'
16    then return Match(S,I+1,J-1)
17    else return Match(S,I+1,J);
18  end;
19 end;
```

Explain how the function works. Your explanation should say what the roles of the parameters `I` and `J` are, and what their values should be when the function is first called. [6]

(c) Write a recursive procedure that computes the maximum level of nesting of brackets in a string. For example, the maximum level of nesting in the string "Happy(Fred)" is 1, while the maximum level of nesting in the string "Loves(Mother(x),x)" is 2. You may assume the string has matching parentheses. [8]

E2.7B

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2003

PRINCIPLES OF COMPUTERS AND SOFTWARE ENGINEERING

Wednesday, 11 June 2:00 pm

Time allowed: 1:30 hours

There are THREE questions on this paper.

Answer TWO questions.

Corrected Copy

This exam is OPEN BOOK

Any special instructions for invigilators and information for candidates are in page 1.

Examiners responsible: First Marker(s) G.A. Constantinides
Second Marker(s): Y.K. Demiris

Special information for invigilators: none

Special information for candidates: The notation "Ox" before a number means that the number is expressed using hexidecimal representation.

The Questions

1.

An ARM program is shown below. (SWI 0x02 prints the null-terminated string at address `r0` to the screen).

```
                AREA prog, CODE, READONLY
SWI_Write0      EQU         0x02
SWI_Exit        EQU         0x11

                ENTRY

                ADR         r0,    buffer
                SWI         SWI_Write0
                BL          Jumble
                SWI         SWI_Write0
                SWI         SWI_Exit

Jumble          STMED       r13!, {r0,r1}
JumbleLoop      LDRB        r1, [r0], #1
                CMP         r1, #0
                RSBNE       r1, r1, #('A'+'Z')
                STRNEB      r1, [r0,#-1]
                BNE         JumbleLoop
                LDMED       r13!, {r0,r1}
                MOV         pc, r14

buffer          =           "MYTEXT", 0

                END
```

a) What function does the subroutine `Jumble` perform? [2]

b) What would you see on the screen if you execute this program? [2]

When assembled, the address of the entry point to this program is 0x1000. Just before execution, `r0` = 0x0, `r1` = 0x0, and `r13` = 0x0.

c) State the value of `r13` during execution of the `Jumble` subroutine. [2]

d) Draw a diagram of the stack during execution of the Jumble subroutine, clearly labelling the addresses and values of all entries. [8]

e) You are required to change the program so that only upper-case letters are modified by the `Jumble` subroutine. All other characters should remain unchanged. Write ARM code for the new version of the subroutine.

[6]

2.

The Fibonacci sequence is: 1, 1, 2, 3, 5, ... and is defined by the following recurrence.

$$F_1 = 1$$
$$F_2 = 1$$
$$F_n = F_{n-1} + F_{n-2}, \text{ for } n > 2$$

Part of an ARM program to generate and print a Fibonacci sequence of length 20 is shown below.

After creating the sequence using a subroutine called `Fibonacci`, the program prints the sequence using a subroutine `PrintNum`, which prints the contents of `r1` to the screen followed by a carriage-return / line-feed combination.

```
                AREA prog, CODE, READONLY
SWI_Exit        EQU    0x11
SeqLength       EQU    20

                ENTRY

                ; Create Fibonacci Sequence
                ADR    r0, buffer
                MOV    r1, #SeqLength
                BL     Fibonacci

                ; Print Sequence
                MOV    r2, r1
PrintLoop       LDR    r1, [r0], #4
                BL     PrintNum
                SUBS   r2, r2, #1
                BNE    PrintLoop

                ; Exit
                SWI    SWI_Exit

; Fibonnacci Sequence Generator
; Input: r0 - pointer to buffer to fill with sequence
;        r1 - desired sequence length (r1 > 2)
; Output: buffer is filled with sequence

        [  SUBROUTINE HERE   ]

buffer          %      4*SeqLength

                END
```

Write the routine `Fibonacci`, to go in the program in the space identified. Comment your code.

[20]

3.

You have been given the task of improving the performance of a computer system. The computer has a 16-bit address bus and a 16-bit data bus, and is byte-addressed (i.e. each byte in memory has a distinct address).

A cache with 8 lines, each of one byte, has been suggested.

In order to evaluate the design, a typical program has been executed, and is found to result in the following memory accesses, ordered by time.

1. Read value 0x01 from address 0x0000
2. Read value 0x01 from address 0x0001
3. Write value 0x02 to address 0x0000
4. Read value 0x02 from address 0x0002
5. Read value 0x03 from address 0x0003
6. Write value 0x06 to address 0x0001
7. Read value 0x05 from address 0x0004
8. Read value 0x08 from address 0x0005
9. Write value 0xA0 to address 0x0002

a) State the principles of spatial and temporal locality

[4]

b) Draw a diagram illustrating the cache contents after the access sequence above has been completed. For each cache line, include the tag, the valid bit, and the data.

[8]

c) State the number of cache misses caused by the above sequence.

[2]

d) An alternative design is to use a 4-line cache with two bytes per line. Which of the memory accesses in the above sequence result in cache misses, when applied to the alternative design?

[6]

Model Answers

1. (a) [New theoretical application]

Iteration 1: List1 = [Chris, Ellen], List2 = [Billy, Darren], List3 = [Amber]

Iteration 2: List1 = [Chris, Ellen], List2 = [Darren], List3 = [Billy, Amber]

Iteration 3: List1 = [Ellen], List2 = [Darren], List3 = [Chris, Billy, Amber]

Iteration 4: List1 = [Ellen], List2 = [], List3 = [Darren, Chris, Billy, Amber]

Iteration 5: List1 = [], List2 = [], List3 = [Ellen, Darren, Chris, Billy, Amber]

(b) [New theoretical application]

The procedure takes two ordered lists List1, and List2, and produces a third list List3, in reverse order, which is the result of merging List1 and List2.

It works by repeatedly taking the head off one of the lists and adding it to the third (which is initialised to be empty). The head of the list chosen is always less-than-or-equal to the head of the other list, thus ensuring the reverse orderedness of the final list.

List1 and List2 must themselves be ordered for the procedure to work correctly.

(c) [New theoretical application]

If the arguments were declared as call-by-reference parameters, then the values of the variables assigned to List1 and List2 in the calling procedure would be modified. This would mean they would be overwritten by empty lists when the function terminated. The function would otherwise still work, however.

2. (a) [New theoretical application]

```
procedure Blur(A1 : TArray, var A2 : TArray);
var X, Y : integer;
begin
  for X := 2 to N-1
    for Y := 2 to M-1
    begin
      A2[X,Y] := A1[X,Y]+A1[X+1,Y]+A1[X,Y+1]+
                 A1[X-1,Y]+A1[X,Y-1];
      A2[X,Y] := A2[X,Y]/5;
    end;
    // Edges
    for X := 1 to N
    begin
      A2[X,1] := A1[X,1];
      A2[X,M] := A1[X,M];
    end;
    for Y := 1 to M
    begin
      A2[1,Y] := A1[1,Y];
      A2[N,Y] := A1[N,Y];
    end;
end;
```

(b) [New theoretical application]

For the main pair of nested for loops, the procedure will execute (N–2)*(M–2)*4 add instructions for the body of the loop, plus (N–2)*(M–2) adds for the inner loop variable, plus (N–2) adds for the outer loop variable. The two "edges" for loops will execute a further N+M add instructions. So the total is (N–2)*(M–2)*5+(N–2)+N+M.

(c) [New theoretical application]

```
function SameShape(T1, T2 : TTree): boolean;
begin
  if T1 = EmptyTree
  then return (T2 = EmptyTree)
  else begin
    F := SameShape(Left(T1),Left(T2));
    F := F and SameShape(Right(T1),Right(T2));
    return F;
  end;
end;
```

3. (a) [New theoretical application]

```
function Match(S : string): boolean;
var I, N : integer;
  Flag : boolean;
begin
  Flag := True;
  N := 0;
  for I := 1 to Length(S)
  begin
    if S[I] = '('
    then N := N+1
    else if S[I] = ')'
    then N := N-1;
    if N < 0 then Flag := False;
  end;
  if N <> 0 then Flag := False;
  return Flag;
end;
```

(b) [New theoretical application]

The function must be called with I = 1 and J = 0. I is an index into the string S, and J is the level of nesting of brackets in S within which the Ith character falls. Every time an open bracket is encountered, J is incremented, and the function is called recursively with the index advanced by one. Similarly, every time a close bracket is encountered, J is decremented. There are three base cases. First, if the end of the string is reached and the level of nesting (J) is zero, then the parentheses match, and the result ir True. Second, if the end of the string is reached and the level of nesting is not zero, then the brackets don't match and the result is False. Third, if the level of nesting goes negative, then a close bracket has been encountered without a matching open bracket before it, and the result is False.

(c) [New theoretical application]

```
procedure Nesting(S : string; I, J : integer;
var Max : integer);
begin
  if J > Max
  then Max := J;
  if I > Length(S)
  then return Max;
  else begin
    if S[I] = '('
    then Nesting(S,I+1,J+1,Max)
    else if S[I] = ')'
    then Nesting(S,I+1,J-1,Max)
    else Nesting(S,I+1,J,Max);
  end;
end;
```

The Answers

Answer 1

a) Jumble replaces each uppercase character in the string by its alphabet-reversed version, i.e. A -> Z, B -> Y, …, Z -> A. The effect on non-uppercase characters has no special interpretation.

[2 marks]

b) The output from this program would be "MYTEXTNBGVCG"

[2 marks]

c) `r13 = 0xFFFFFFF8`

[2 marks]

d)

Address	Data
0x00000000	0x00000000
0xFFFFFFFC	0x00001034

[8 marks]

e)

```
Jumble          STMED     r13!, {r0,r1}
JumbleLoop      LDRB      r1, [r0], #1
                CMP       r1, #0
                BE        Done
                CMP       r1, #'A'
                BLO       JumbleLoop
                CMP       r1, #'Z'
                BHI       JumbleLoop
                RSB       r1, r1, #('A'+'Z')
                STRNEB    r1, [r0,#-1]
                BNE       JumbleLoop
Done            LDMED     r13!, {r0,r1}
                MOV       pc, r14
```

[6 marks]

Answer 2

```
Fibonacci       STMED   r13!, {r0-r4}
                MOV     r2, #1              ; )
                STR     r2, [r0], #4        ; ) starts with 1,1,
                MOV     r3, #1              ; )
                STR     r3, [r0], #4        ; )

                SUB     r1, r1, #2          ; number of items left to calculate

FibLoop         ADD     r4, r2, r3          ; ) calculate and store next number
                STR     r4, [r0], #4        ; )

                MOV     r3, r2              ; ) advance to next operands
                MOV     r2, r4              ; )

                SUBS    r1, r1, #1          ; ) end of for loop
                BNE     FibLoop             ; )

                LDMED   r13!, {r0-r4}
                MOV     pc, r14
```

[20 marks]

Mark breakdown:

Pushing and popping… [2 marks]
… the appropriate registers [2 marks]
Correctly labelling [2 marks] and returning from [2 marks] the subroutine
Correctly constructing a loop, with appropriate loop bound [4 marks]
Only requiring one memory access per loop iteration [2 marks]
Other correct functioning [6 marks]

A direct recursive implementation is exponential-time & therefore does not deserve full marks. In this case, do not give marks for correct loop construction and loop bound, but no other penalty.

Answer 3

a)

Spatial Locality: If an item is referenced, nearby items are likely to be referenced

Temporal Locality: If an item is referenced, it is likely to be referenced again soon

[4 marks]

b)

Cache Line	Valid	Tag	Data
0x0	Y	0x00	0x02
0x1	Y	0x00	0x06
0x2	Y	0x00	0xA0
0x3	Y	0x00	0x03
0x4	Y	0x00	0x05
0x5	Y	0x00	0x08
0x6	N	-	-
0x7	N	-	-

[8 marks]

c) 6 misses (all the reads).

[2 mark]

d) 3 misses (1st read of every pair)

1. 0x0000 => byte 0, line 0, tag 0, miss
2. 0x0001 => byte 1, line 0, tag 0, hit
3. 0x0000 => byte 0, line 0, tag 0, hit
4. 0x0002 => byte 0, line 1, tag 0, miss
5. 0x0003 => byte 1, line 1, tag 0, hit
6. 0x0001 => byte 1, line 0, tag 0, hit
7. 0x0004 => byte 0, line 2, tag 0, miss
8. 0x0005 => byte 1, line 2, tag 0, hit
9. 0x0002 => byte 0, line 1, tag 0, hit

[6 marks]