HL Unit 5 – Abstract Data Structures

Recursion Quiz 1

Question 1			
Objectives:	5.1.1	Exam Reference:	May-17 10

Outline the reason why recursive solutions can be memory intensive. [2]

Award up to ***[2 max]***.
A recursive call involves the use of stacks;
For storing/pushing on/popping out data/ return addresses/return values *etc*;
If many recursive calls are made, the memory usage can be very large;

Question 2			
Objectives:	5.1.1, 5.1.2	Exam Reference:	Nov-16 6

Consider the following recursive algorithm *FUN(X, N)*, where *X* and *N* are two integers.

```
FUN(X, N)
if N<=0 then
  return 1
else
  return X*FUN(X, N-1)
end if
```

The *return* statement gives the value that the algorithm generates.

(a) Determine how many times multiplication is performed when this algorithm is executed. [1]

N;

(b) Determine the value of *FUN(2,3)*, showing all of your working. [3]

```
FUN(2,3)=
=2 * FUN(2,2);
=2 * 2*FUN(2,1);
=2 * 2 * 2 * FUN(2,0);
=2*2*2*1= 8;
```

(c) State the purpose of this recursive algorithm. [1]

Calculates X^N; ***Note:*** *DO NOT accept vague answers that may suggest the understanding of* N^X *or use incorrect terminology*

Question 3			
Objectives:	5.1.1, 5.1.2, 5.1.3	Exam Reference:	May-16 15

1. The letters $F_0, F_1, F_2, \ldots, F_N, \ldots,$ where N ≥ 0, are used to identify the *N th* term of the sequence of Fibonacci numbers that starts as follows.

   0 , 1 , 1 , 2 , 3 , 5 , 8 , 13 , …

   With the exception of the leading 0 and 1 (the zeroth term and 1st term), the terms in the sequence are the sum of the two preceding terms. For example, F_5 is the 5th term of the sequence, which is 5, and is the sum of the 3rd and 4th terms, which are 2 and 3 respectively.

   (a) State the value of the 8th term in the sequence. [1]

21

The following method, *fibo(N)*, generates the *N th* term in the sequence. The *return* statement returns the value that the method generates.

```
fibo(N)
  if (N=0 OR N=1) then
    return N
  else
    return (fibo(N-1) + fibo(N-2))
  end if
```

(b) Trace *fibo(4)*, showing the different levels of recursion. [3]

*Award marks as follows up to **[3 max]**.*
*Award **[1]** for evidence of **two recursive calls** of `fibo(2)` or of two recursive calls of `fibo(0)`.*
*Award **[1]** for evidence of **three recursive calls** of `fibo(1)`.*
*Award **[1]** for correctly returning values `0` and `1` for `fibo(0)` and `fibo(1)` when needed.*

For example:

```
fibo(4)
= (fibo(3)+fibo(2))
= (fibo(2)+fibo(1)) + (fibo(1)+fibo(0))
= ((fibo(1)+fibo(0)) +1) + (1+0)
= ((1+0)+1) + (1+0)
= 3
```

***Note:** No marks shall be given if the answer 3 is provided without any tracing, nor if the intermediate values instead of calls to `fibo()` are given; this is a tracing question.*
The order of evaluation of intermediate steps may be different from the one presented in the solution here.

(c) Construct a non-recursive algorithm to generate Fibonacci numbers. [6]

*Award marks as follows, up to **[6 max]**.*
*Award **[1]** for initialization.*
*Award **[1]** for correct condition in If statement.*
*Award **[1]** for correct loop within if statement.*
*Award **[1]** for correctly summing up.*
*Award **[1]** for correct swap of variables in the loop.*
*Award **[1]** for correct return/output.*

Example 1:

```
fibo(N)
if N==0 OR N=1
  RES = N      // RES stores the result
else
  BACK2 = 0 // Initialize variables
  BACK1 = 1
  loop J from 2 to N
    RES = BACK1 + BACK2
    BACK2 = BACK1
    BACK1 = RES
  end loop
end if
return RES
```

Example 2:

```
fibo(N)
V = 1     // Initialize variables
RES = 1   // RES stores the result
if (N=0) then
  return N
else
  loop J from 3 to N
    TEMP = V + RES
    V = RES
    RES = TEMP
  end loop
end if
return RES
```

(d) Construct an algorithm that will **output** the first *N* terms of the sequence. You should use *fibo()*, the method defined above. [3]

*Award marks as follows, up to **[3 max]**.*
*Award **[1]** for setting up a loop.*
*Award **[1]** for calling* `fibo()`.
*Award **[1]** for outputting the result.*

Example 1:

```
procedure(N)
loop J from 0 to N
  output(fibo(J))
end loop
```

Example 2:

```
procedure(N);
arrayfibo[N]   // declare array with N positions
loop J from 0 to N
  arrayfibo[J] = fibo(J)
end loop
H = 0
loop while H=<N
  output arrayfibo[H];
    H = H+1;
end loop
```

Recursive programs written in high level languages require the use of particular structures to support their execution.

(e) Describe how a stack is usually employed in the running of a recursive algorithm. [2]

The current environment (*eg* values/local variables/current address/registers) PUSHED onto the stack when a new recursive call is met;

To be POPPED OFF the stack when the recursive subprogram is completed.

Question 4			
Objectives:	5.1.1, 5.1.2	Exam Reference:	Nov 15 10.f

Consider the following **recursive** algorithm, in which *X* and *Y* are parameters in the method *F*. The *return* statement gives the value that the method generates.

```
F(X,Y)
    if X < Y then
        return F(X+1,Y-2)
    else if X = Y
        return 2*F(X+2,Y-2)-2
    else
        return 2*X+4*Y
    end if
```

(f) Determine the value of F(5,11). [5]

Award [1] for each correct line.

```
F(5,11) = F(6,9)
        = F(7,7)
        = 2 * F(9,5) - 2
        = 2*(2*9 + 4*5) -2
        = 2 * 38 -2 = 74
```