

APOLLO

B1 Computational thinking

HL

Computer Science
First assessment 2027
4 questions | mark schemes

1. (a) *Award [1 max]*

because it is in diagrammatic form;
 it is program agnostic;
 no complicated code / syntax to follow;

(b) *Award [3]*

correct values of i;
correct values of j;
correct output;

i	i <= N ?	j	j <= N ?	output
1	TRUE			1
		2	TRUE	2
		3	TRUE	3
		4	FALSE	
2	TRUE			2
		3	TRUE	3
		4	FALSE	
3	TRUE	4	FALSE	3
4	FALSE			

(c) *Award [1]. Note: the algorithm shows a nested loop.*

$O(n^2)$;

– 6 –

(d) Award **[5 max]** as follows

opening *employees* file for reading / default mode;
open *surnames* file for writing;
looping through *employees* file;
use of a counter or equivalent;
writing surnames only to *surnames* file;
closing both files;

*Note: Award **[1]** for first 2 marking points if files are opened without indicating access modes.*

Algorithm is:

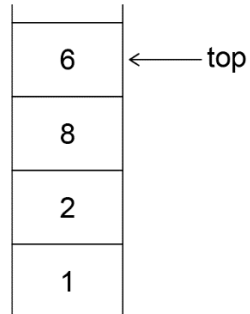
```
open employees file for reading / in default mode
open surnames file for writing
set counter to 0
repeat until end of employees file
    read next line from employees file
    if counter%3 = 0 *
        write line to surnames file
    increment counter
close both files
```

* there are alternatives ways of using counter

– 7 –

(e) *Award [2].*

correct elements in stack **[1]**;
correct indication of top **[1]**;



(f) *Award [3] as follows:*

Award [1] for each correct line (or equivalent).

repeat until isEmpty is true / while isEmpty is false;

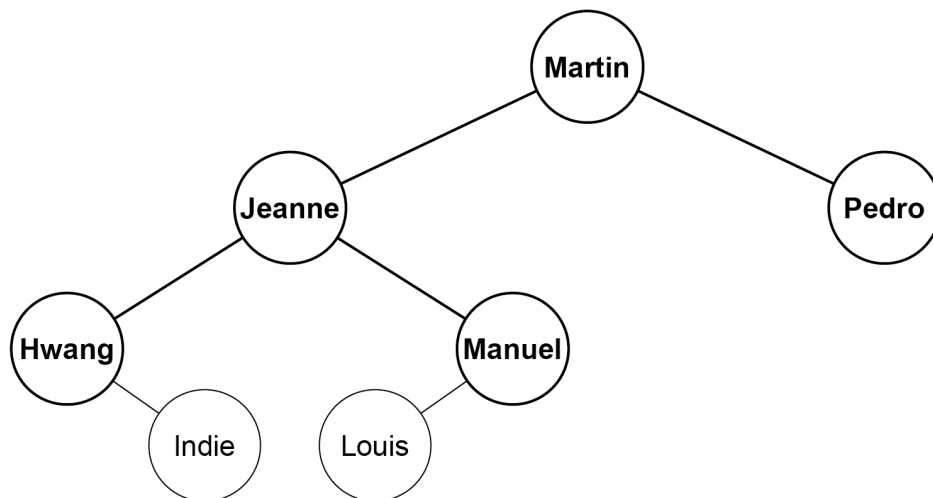
pop an element and save its value/save in a variable/push it onto a new stack (stack2)

print the last value saved/print the variable/pop stack2 and print value

Note: Answer must demonstrate an algorithmic approach.

5. (a) **Award [2 max]**

award [1] for each correct entry;

(b) **Award [3 max]**

The insert method is initially passed the root of the tree as a parameter (where a comparison is made);

Then either the left or right subtree is selected (depending upon the result of the comparison);

The method is called again, using the root of this subtree as the parameter;

This is repeated until the insertion point is found (base case) / subtree is null;

(c) (i) **Award [1 max]**

21;

(ii) **Award [4 max]**

Award [1] for including a recursive call that will lead to the base case;

Award [1] for having a base case;

Award [1] for return + int;

Award [1] for a completely correct algorithm;

```

public int recur(int x)
{
    if (x != 0)
        return x + recur(x-1);
    else
        return x;
}
  
```

(iii) **Award [3 max]**

data is stored in the memory / on a stack every time a recursive call is made;

if the value of x is very large ...;

... the stack may overflow / system may run out of memory;

7. (a) **Award [1 max]**

because it is in diagrammatic form;
it is program agnostic;
no complicated code / syntax to follow;

(b) **Award [3]**

correct values of i ;
correct values j ;
correct output;

i	$i \leq N ?$	j	$j \leq N ?$	output
1	TRUE			1
		2	TRUE	2
		3	TRUE	3
		4	FALSE	
2	TRUE			2
		3	TRUE	3
		4	FALSE	
3	TRUE	4	FALSE	3
4	FALSE			

(c) **Award [1]. Note: the algorithm shows a nested loop.**

$O(n^2)$;

– 15 –

(d) Award **[5 max]** as follows

opening *employees* file for reading / default mode;
open *surnames* file for writing;
looping through *employees* file;
use of a counter or equivalent;
writing surnames only to *surnames* file;
closing both files;

*Note: Award **[1]** for first 2 marking points if files are opened without indicating access modes*

Algorithm is:

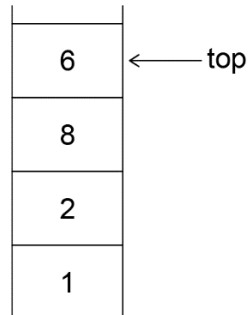
```
open employees file for reading / in default mode
open surnames file for writing
set counter to 0
repeat until end of employees file
    read next line from employees file
    if counter%3 = 0 *
        write line to surnames file
    increment counter
close both files
```

* there are alternatives ways of using counter

– 16 –

(e) *Award [2].*

correct elements in stack **[1]**;
correct indication of top **[1]**;



(f) *Award [3] as follows:*

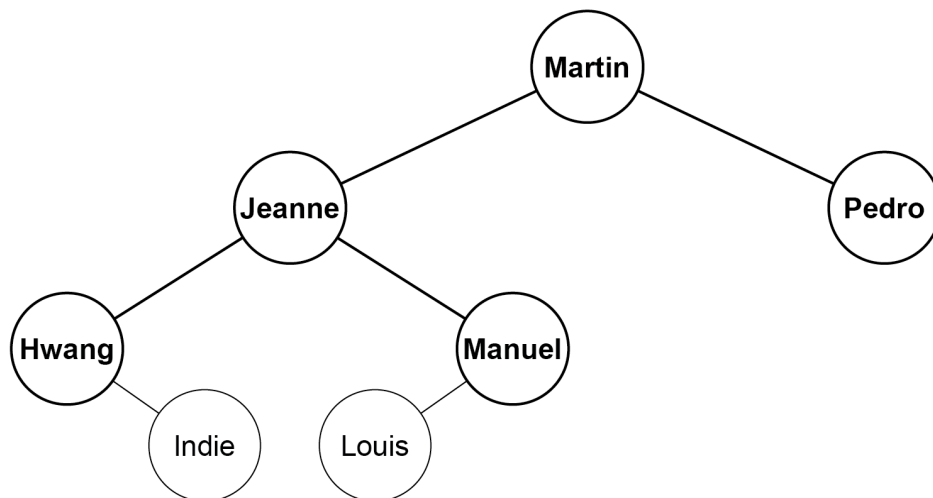
Award [1] for each correct line (or equivalent).

repeat until isEmpty is true / while isEmpty is false;
 pop an element and save its value/save in a variable/push it onto a new stack (stack2);
print the last value saved/print the variable/pop stack2 and print value;

Note: Answer must demonstrate an algorithmic approach.

11. (a) **Award [2 max]**

award [1] for each correct entry;

(b) **Award [3 max]**

The insert method is initially passed the root of the tree as a parameter (where a comparison is made);

Then either the left or right subtree is selected (depending upon the result of the comparison);

The method is called again, using the root of this subtree as the parameter;

This is repeated until the insertion point is found (base case) / subtree is null;

(c) (i) **Award [1 max]**

21;

(ii) **Award [4 max]**

Award [1] for including a recursive call that will lead to the base case;

Award [1] for having a base case;

Award [1] for return;

Award [1] for a completely correct algorithm;

```

def recur(x):
    if x != 0:
        return x + recur(x-1)
    else:
        return x
  
```

(iii) **Award [3 max]**

data is stored each in the memory / on a stack every time a recursive call is made;

if the value of x is very large ...;

... the stack might overflow / system might run out of memory;