Cambridge International AS & A Level

Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

COMPUTER SCIENCE **9608/43**

Paper 4 Further Problem-solving and Programming Skills **May/June 2019**

PRE-RELEASE MATERIAL

5803001003

No Additional Materials are required.

This material should be given to the relevant teachers and candidates as soon as it has been received at the centre.

READ THESE INSTRUCTIONS FIRST

Candidates should use this material in preparation for the examination. Candidates should attempt the practical programming tasks using their chosen high-level, procedural programming language.

DC (KS/CB) 163591/4

Teachers and candidates should read this material prior to the June 2019 examination for 9608 Paper 4.

Reminders

The syllabus states:

- there will be questions on the examination paper which do not relate to this pre-release material
- you must choose a high-level programming language from this list:
    - Visual Basic (console mode)
    - Python
    - Pascal / Delphi (console mode)

Note: A mark of **zero** will be awarded if a programming language other than those listed is used.

The practical skills for Paper 4 build on the practical skills covered in Paper 2. We therefore recommend that candidates choose the same high-level programming language for this paper as they did for Paper 2. This will give candidates the opportunity for extensive practice and allow them to acquire sufficient expertise.

Questions on the examination paper may ask the candidate to write:

- structured English
- pseudocode
- program code

A program flowchart should be considered as an alternative to pseudocode for documenting a high-level algorithm design.

Candidates should be confident with:

- the presentation of an algorithm using either a program flowchart or pseudocode
- the production of a program flowchart from given pseudocode and vice versa

Candidates will also benefit from using pre-release materials from previous examinations.

Declaration of variables

The syllabus document shows the syntax expected for a declaration statement in pseudocode.

```
DECLARE <identifier> : <data type>
```

If Python is the chosen language, each variable's identifier (name) and its intended data type must be documented using a comment statement.

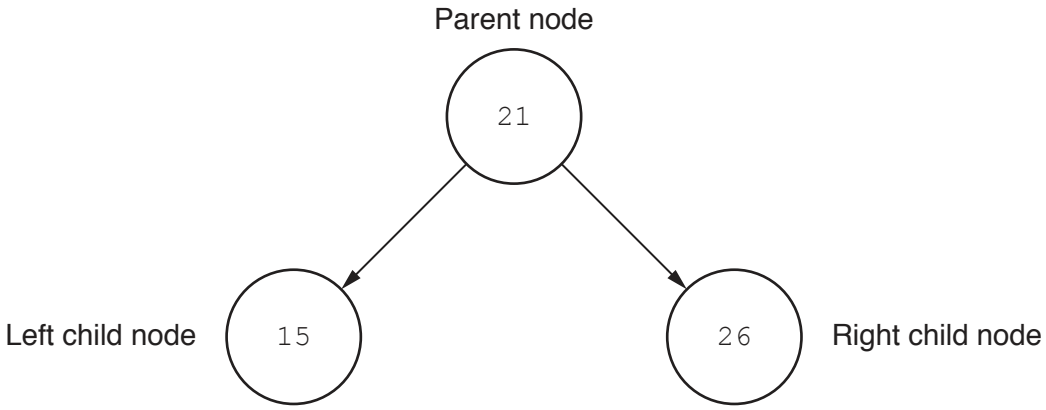
Structured English – Variables

An algorithm in pseudocode uses variables, which should be declared. An algorithm in structured English does not always use variables. In this case, the candidate needs to use the information given in the question to complete an identifier table. The table needs to contain an identifier, data type and description for each variable.

TASK 1 - Binary trees and linked lists

TASK 1.1

A binary tree is a type of data structure. In a binary tree, each node will have up to two connected nodes. These connected nodes are called child nodes. Each parent node can have a left child node and a right child node. For example:



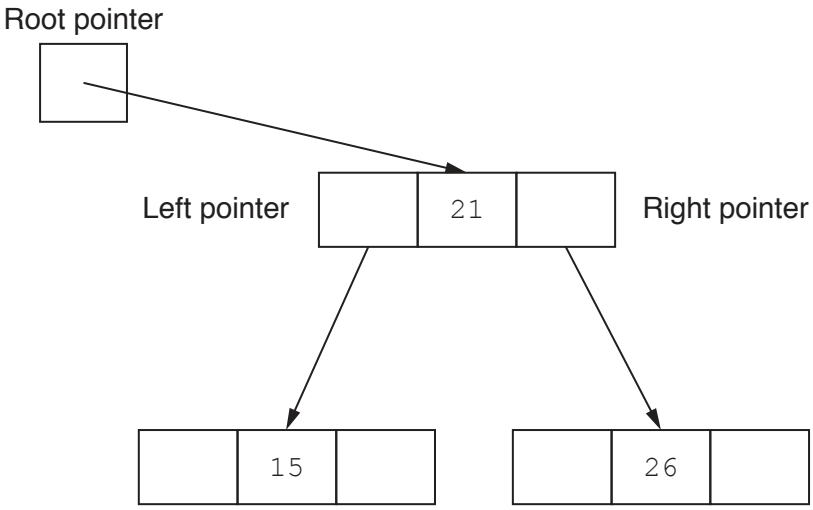
Add the data 10, 34 and 22 to the binary tree.

Create a random list of 10 items of data about your favourite hobby or topic, for example, 10 different types of bird.

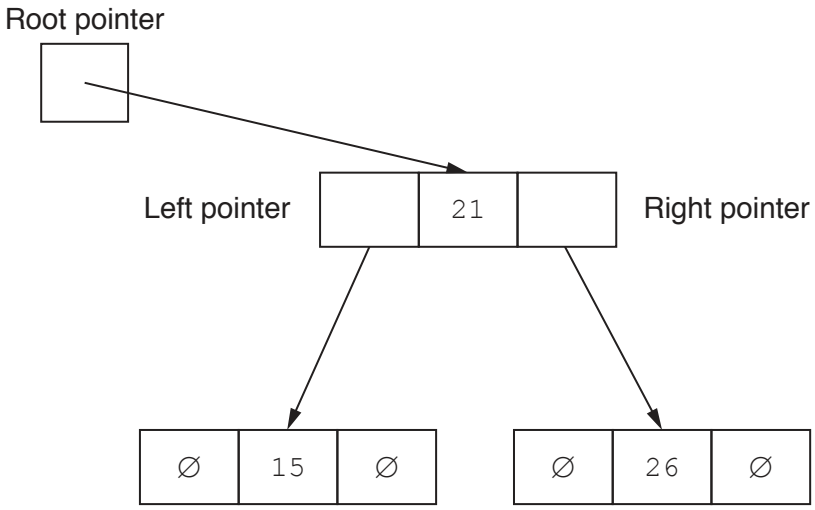
Create a binary tree for the list of items. Add items to the tree in ascending alphabetical order, `'a'` to `'z'`.

TASK 1.2

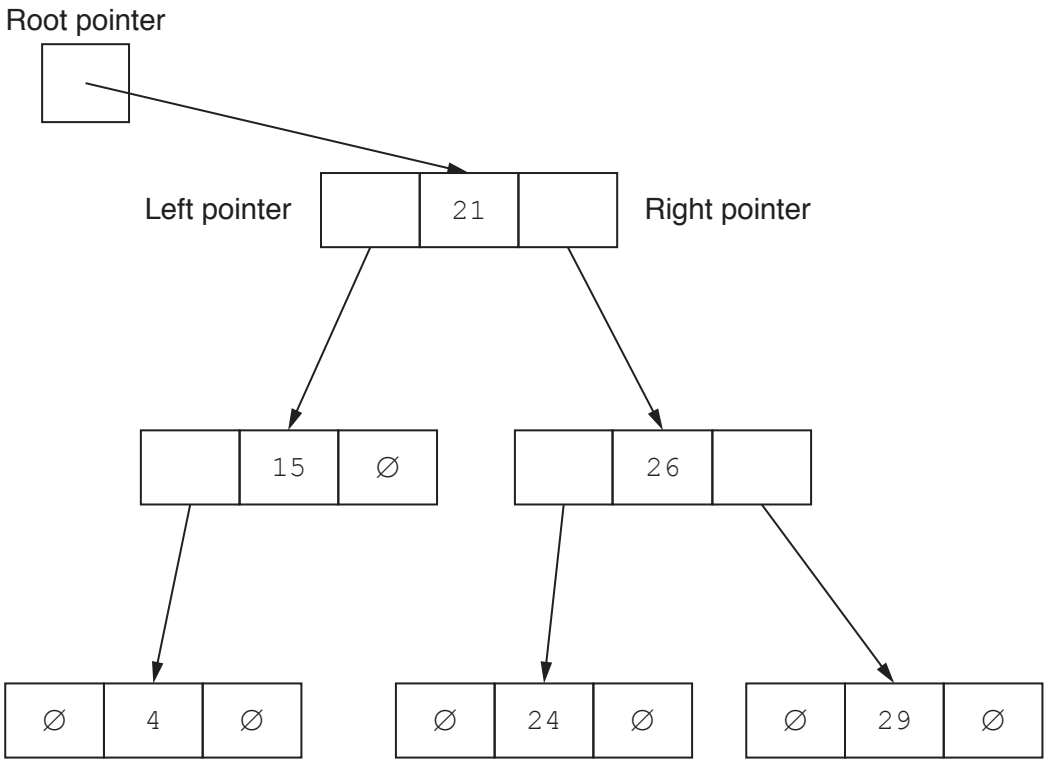
It is possible to represent data in a binary tree as a linked list of nodes. Each node can have a left pointer and a right pointer. For example:



Whenever a node does not have a child node, the relevant pointer stores a null symbol, such as the symbol ∅. For example, in our previous tree that has two childless, leaf nodes:



If we add three more items to the tree, it becomes:



In **TASK 1.1**, you added your 10 items to a binary tree. Now redraw your tree as a linked list of nodes. Use the correct pointer value wherever the node has no child.

TASK 1.3

The data in the linked list binary tree at the end of **TASK 1.2** are suitable for storage in a 2D array.

We can represent the 2D array for the numerical data as a table. For example:

RootPointer

0

FreePointer

	LeftPointer	Data	RightPointer
0		21	
1		15	
2		26	
3		4	
4		24	
5		29	
6			

The left and right pointer can be completed in the table. The other pointers are completed by assigning the right pointer of the node first, and then left. The free pointer is then assigned the position of the next available array index. For example:

RootPointer

0

FreePointer

6

	LeftPointer	Data	RightPointer
0	1	21	2
1	3	15	∅
2	4	26	5
3	∅	4	∅
4	∅	24	∅
5	∅	29	∅
6			

Create a table to represent an array of records for your linked list binary tree, completing the left, right and free pointers.

TASK 1.4

Research how an item of data is added to a linked list.

Create a program in the language of your choice that adds an item of data to a list.

[Turn over

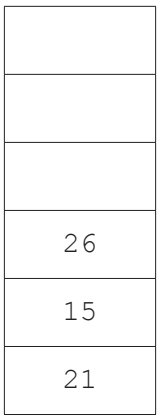
TASK 2 - Stacks

A stack is an Abstract Data Type (ADT) that stores a collection of data. It has two operations that manipulate its data:

- `PUSH`, which adds an item to the collection
- `POP`, which removes the item that the program most recently added unless the stack is already empty.

TASK 2.1

The values `21`, `15` and `26` are currently stored in a stack as follows:



The following commands are carried out.

```
PUSH(4)

POP()

POP()

PUSH(24)

PUSH(29)
```

Draw a representation to show what the stack would look like after the commands have been carried out.

TASK 2.2

Write a program in the language of your choice to add and remove items of data from a stack.

BLANK PAGE

BLANK PAGE