Year 12-13 Building Work
Cover Sheet

SUBJECT: Computer Science	**DATE: Summer 2026**

The purpose of departmental building work is to provide students with a series of engaging tasks that aim to ease transition from Years 12-13 and ensure progress post- Year 12 examinations.

The aims of this Building Work are:

1. To maintain the foundation of the Year 12 learning in Computer Science.
2. To develop and support your understanding of the Tier 2 terminology for OCR Computer Science with a view to achieving higher marks.
3. To kickstart the NEA process so that the initial planning and feasibility has been checked before the start of Year 13.

INSTRUCTIONS:

Section A: Consolidation Exercises

1. Consolidate your understanding of virtual machines. Complete task 1.
2. Consolidate your understanding of functions and procedures. Complete task 2.
3. Consolidate your understanding of data structures. Complete task 3

Section B: Extended response question practice

Complete task 4

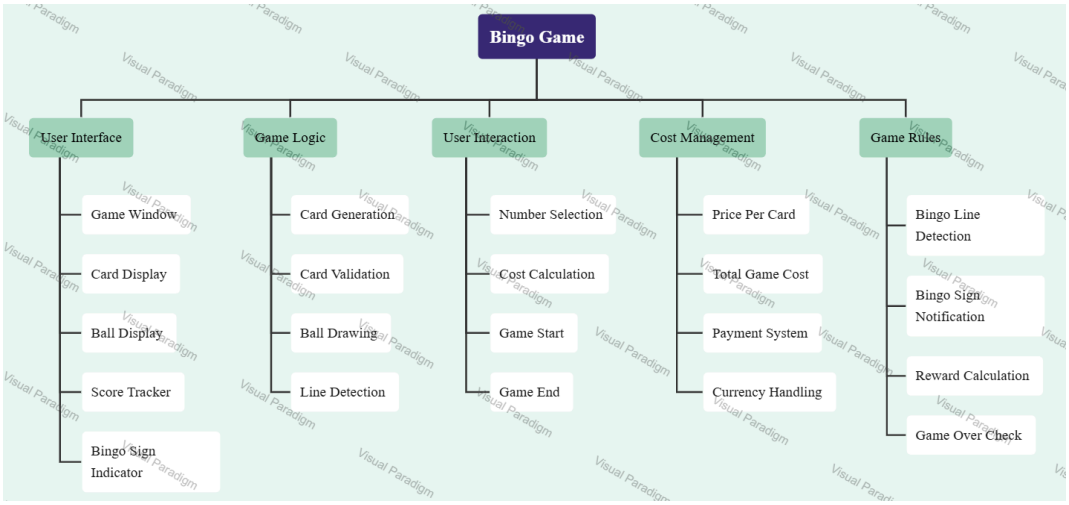
Section C: Resits (Some students)

You will be told if you need to complete this section by your teacher.

Section D: NEA Analysis Writeup

You should have the first complete draft of your Analysis by the time you arrive back in September. To ensure that your project is feasible, you must also plan out:

- what the final objectives will be and how you will know if you have been successful or not
- a decomposition diagram of the parts of your project, e.g.



- a timeline of how many hours you will use for each part of the program to make sure you know your scope is achievable. You can investigate GANTT charts for suggestions on layouts.

FOLLOW UP:

Your building work will be assessed in the following way/s:

Section A: Consolidation tasks

Task 1: Short exam question in class on return-peer marked with whole class feedback. This will focus on A01 and A02 points.
Task 2: Smart revise questions on return to test retrieval
Task 3: Pseudo-code exercise on return that will be peer marked.

Section B: Extended answer response

Teacher assessed focussing on A01, A02 and A03 terminology.

Section C: resits

Resit Exam Teacher-assessed after resit period

Section D: NEA

Teacher assessed with whole class general feedback.

Section A: Consolidation Exercises

Task 1: Virtual Machines

A. Answer the following questions:

1. What is a virtual machine?
2. Why is a virtual machine described as software taking on the function of a machine?
3. Give one example of a virtual machine used to run an operating system.
4. What is intermediate code?
5. Give one example of a virtual machine used to execute intermediate code.
6. Why might a programmer choose to use a virtual machine?
7. Can a virtual machine exist without pretending to be a whole physical computer? Explain briefly.
8. What is the difference between hardware and the software layer in a virtual machine system?

B. Complete the paragraph using the words below.

Words:

virtual machine, intermediate code, environment, operating system, portability, software, host, guest, machine, bytecode

A _____ is a layer of _____ that takes on the function of a _____. It may allow a _____ operating system to run inside a _____ system, or it may provide an execution _____ for _____ such as Java _____. One advantage is improved _____, because the same program can run wherever the same VM is available.

C. Misconception Check

For each statement:

1. Decide whether it is **True** or **False**.
2. If false, rewrite it so it becomes correct.

Statement	True/False	Corrected version
1. A virtual machine is a physical computer.		
2. A virtual machine can run its own operating system.		
3. A virtual machine must have dedicated hardware.		
4. Java programs run on a virtual machine called the JVM.		
5. The host operating system is installed inside the virtual machine.		
6. A virtual machine allows one computer to behave like multiple computers.		
7. Virtual machines always perform faster than physical machines.		
8. Intermediate code can be executed by a virtual machine.		
9. The JVM translates Java bytecode into machine code.		
10. Virtual machines are only used for programming languages.		

Task 2: Functions and Procedures

A. Explain the difference between a Function and Procedure

Complete the table

Feature	Function	Procedure
Returns a value?	Yes/No	Yes/No
Can be used in calculations?	Yes/No	Yes/No
Main purpose		

B. Explain the difference between Pass-by-Value and Pass-by-Reference

Complete the table

Feature	Function	Procedure
Original variable modified?	Yes/No	Yes/No
Copy created?	Yes/No	Yes/No
Main purpose		

C. Tracing Pseudocode

Study the code:

```
FUNCTION MultiplyByTwo(number)
    result ← number * 2
    RETURN result
ENDFUNCTION

value ← 6
answer ← MultiplyByTwo(value)
OUTPUT value
OUTPUT answer
```

Answer the questions:

1. What is output first?
2. What is output second?
3. Does the value stored in value change?
4. Why does this demonstrate passing by value?

D. Write the ppseudocode and test in OCR Exam Reference Language Interpreter

1. Write pseudocode for a **procedure** called DisplayLine that:
    - accepts one parameter called text
    - outputs that text

2. Write a **function** called IsEven that:
    - accepts one integer
    - returns TRUE if it is even
    - otherwise returns FALSE

3. Write a short pseudocode program that:
    - stores score ← 8
    - calls a procedure that adds 10 **by value**
    - outputs the score afterwards
    - shows that the original score has not changed

4. Write a short pseudocode program that:
    - stores score ← 8
    - calls a procedure that adds 10 by reference
    - outputs the score afterwards
    - shows that the original score has changed

Task 3: Data structures

A. Higher Order Questions to answer

1. Why is an array often efficient for direct access but less flexible when the size changes?
2. Why might a linked list be preferred over an array in some situations?
3. Why is a stack suitable for function calls or undo operations?
4. Why is a queue suitable for scheduling and buffering?
5. Why are graphs useful for modelling real-world systems?
6. Why can a binary search tree provide faster searching than an unsorted list?
7. Why can collisions reduce the efficiency of a hash table?
8. Why is choosing the correct data structure important in algorithm design?

B. Misconception Check

For each statement in the table on the next page:

1. Decide whether it is **True** or **False.**
2. If false, rewrite it so it becomes correct.

Statement	True/ False	Corrected version – explain why the statement is false if chosen, and expand on the statement if true
1. A list and an array are always exactly the same thing.		
2. Arrays store items in contiguous memory locations.		
3. A tuple is generally used to store a fixed collection of items.		
4. Records store multiple values about one entity using named fields.		
5. A stack uses FIFO.		
6. A queue uses LIFO.		
7. A linked list stores items in adjacent memory locations.		
8. A binary search tree always has exactly two children per node.		
9. In a directed graph, edges have direction.		
10. A tree is a special type of graph.		
11. A hash table stores values using a hash function and key.		

12. A graph must have a root node.		
13. Arrays can have more than one dimension.		
14. A record is best described as a sequence of unnamed values.		
15. A binary search tree is ordered.		

Section B:

Answer the question

* ‘Technology is changing too quickly for the law to keep up.’

Discuss to what extent you agree with the statement above. In your discussion you should explain which laws regulate the use of technology and how advancements in technology have made the laws difficult to enforce/implement. [12 mark]

Section C: Resit

Preparation for resit exam

1.1 Components of a computer and their uses

1.1.1 Structure and function of the processor

1.1.2 b. GPUs
c. multicore/parallel systems

1.1.3 Input, output and storage

1.2 Software and software development

1.2.1 Systems Software

1.2.2 Applications Generation
b. Utilities

1.2.3 Types of Programming Language
e. Object-oriented Languages

1.4 Data Types, data structures and algorithms

1.4.1 Data Types

a. Primitive data types
b. positive numbers in binary
c. use of sign and magnitude, two's complement for negative numbers
d. addition and subtraction of binary numbers

1.4.2 Data Structures [procedural or OOP]

a. arrays, records, lists, tuples
b. linked-lists, tree (create and depth traverse only), stack, queue and hash tables
c. How to create, traverse, add to and remove from the data structures above. (See trees)

2.1 Elements of computational thinking

2.1.1 Thinking abstractly
2.1.2 Thinking ahead
2.1.3 Thinking procedurally
2.1.4 Thinking logically
2.1.5 Thinking concurrently

2.2 Problem solving and programming

2.2.1 Programming techniques

Other skills

Pseudocode: reading, tracing and writing