OCR Computer Science AS Level

1.4.2 Data Structures

Concise Notes

Specification

1.4.2 a)

- Arrays
- Records
- Lists
- Tuples

1.4.2 b)

- Stack
- Queue

Arrays, Records, Lists, and Tuples

Arrays

- An array is an ordered, finite set of elements of a single type.
- A 1D (one- dimensional) array is a linear array.
- A 2D (two-dimensional) array can be visualised as a table/spreadsheet.
- When searching an array, first go down the rows and then across the columns.
- A 3D (three-dimensional) array can be visualised as a multi-page spreadsheet
    - An element in a 3D array using: `threeDimensionalArray[z,y,x]`
      z = array number, y = row number, x = column number.

Records

- More commonly referred to as a row in a file,
- A record is made up of fields, and is widely used in databases.

Lists

- Consists of a number of items, where items can occur more than once.
- Data can be stored in non-contiguous locations and be of more than one data type.

Tuples

- An ordered set of values of any data type .
- Cannot be changed: elements cannot be added, edited or removed once initialised.
- Initialised with regular brackets rather than square brackets

Stacks and Queues

Stacks

- Last in first out (LIFO) data structure:
    - Items can only be added to/ removed from the top of the stack.
- Used to reverse actions, eg. back buttons and undo buttons use stacks
- Can be implemented as a static or dynamic structure.

Queues

- First in first out (FIFO) data structure:
    - Items are added to the end and are removed from the front of the queue.
- Used in printers, keyboards and simulators.
- Linear queue: items are added into the next available space, starting from the front.
    - Items are removed from the front of the queue
    - Uses two pointers: pointing to the front and back of the queue.

  - Use space inefficiently, as positions from which data has been removed cannot be reused
- Circular queues have a rear pointer that can loop back to the front of the queue and utilise empty space at the front.
  - Are harder to implement.

Operations on Stacks and Queues

Stacks

Stack Operations	Description
isEmpty()	Checks if the stack is empty
push(value)	Adds a new value to the top of the stack
peek()	Returns the top value of the stack without removing it
pop()	Returns and removes the top value of the stack
size()	Returns the size of the stack
isFull()	Checks if the stack if full

Queues

Queue Operations	Description
enQueue(value)	Adds a new item to the end of the queue
deQueue()	Removes the item from the end of the queue
isEmpty()	Checks if the queue if empty
isFull()	Checks if the queue is full