NAME DATE STARTED TARGET END DATE

GCE A LEVEL — COMPUTER SCIENCE UNIT 3 QUESTION PACK

1500U30-1 · 2015 spec Unit 3 Topic 3 · A2 unit, first sat 2017, 100 marks, 2h paper

REVISE.wales

COMPUTER SCIENCE — UNIT 3 · Algorithms — Sorts, Searches, Recursion & Big-O Complexity

Topic 3.3 — Algorithm design, dry-running, recursion vs iteration, time complexity and growth rates

Writing bubble, quick and merge sort algorithms in pseudo-code, designing iterative and recursive solutions, dry-running algorithms with trace tables, debugging flowchart errors, and analysing time/space complexity in Big-O notation including O(1), O(log n), O(n), O(n²) and O(n³) growth.

2015 specification · current

Estimated time for entire question pack: ~4 h

Derived from the Unit 3 pace of ~1.5 min/mark, padded for written-prose answers (160 marks over 20 questions).

*You are advised to **not** attempt to complete all of this in one sitting.*

ABOUT THIS QUESTION PACK

This is a **comprehensive topic question pack**, not a single mock paper. It contains every question from the WJEC A2 Unit 3 papers (Summer 2017 – Summer 2024, COVID gap) that maps onto Topic 3.3 of the 2015 specification.

Questions are ordered by source paper date.

INSTRUCTIONS

Use black ink or black ball-point pen. Show all working. A calculator is allowed where useful.

For Examiner's use only

Q	Source	Max	Mark
1	S17 Q6	9	
2	S17 Q8	4	
3	S17 Q9	6	
4	S18 Q6	6	
5	S18 Q9	8	
6	S18 Q10	13	
7	S19 Q5	8	
8	S19 Q9	10	
9	S19 Q10	6	
10	S19 Q12	7	

Q	Source	Max	Mark
11	S22 Q2	8	
12	S22 Q4	8	
13	S22 Q5	8	
14	S22 Q11	9	
15	S23 Q5	8	
16	S23 Q12	9	
17	S24 Q4	8	
18	S24 Q5	8	
19	S24 Q9	8	
20	S24 Q11	9	
Total		**160**	

Algorithms — Sorts, Searches, Recursion & Big-O Complexity — what the spec asks

WJEC GCE A Level Computer Science (from 2015) · Unit 3: Programming & System Development · Topic 3.3.

Algorithm definition

- Finite sequence of unambiguous steps that solves a problem.
- Has input(s), output(s), is deterministic, and terminates.
- Can be expressed in pseudo-code, flowchart, structured English or programming language.
- Same problem may have many algorithms with different efficiency.

Iteration vs recursion

- Iteration: explicit loops (FOR / WHILE / REPEAT) — uses constant stack space.
- Recursion: function calls itself with smaller input until a base case.
- Recursive solutions are elegant for divide-and-conquer (quicksort, tree walks).
- Iterative solutions are usually faster and avoid stack-overflow risk on large inputs.

Sorting algorithms

- Bubble sort: swap adjacent out-of-order pairs; $O(n^2)$ worst case.
- Insertion sort: shift each element into the sorted prefix; $O(n^2)$ worst case.
- Quicksort: pick pivot, partition, recurse; O(n log n) average, $O(n^2)$ worst.
- Merge sort: split, recurse, merge; O(n log n) guaranteed.

Searching algorithms

- Linear search: scan from start; works on unsorted data; O(n).
- Binary search: requires sorted array; halve search range each step; O(log n).
- Maintain `first`, `last`, `mid` pointers; terminate on match or first > last.
- Binary search is dramatically faster: 1,000,000 items in ~20 steps.

Big-O complexity

- O(1) — constant time; e.g. array index lookup.
- O(log n) — logarithmic; e.g. binary search.
- O(n) — linear; e.g. one pass over a list.
- O(n log n) — e.g. quicksort, merge sort average.
- $O(n^2)$, $O(n^3)$ — nested loops over the data.

Dry-running & debugging

- Trace table: one column per variable + outputs; one row per iteration.
- Step line-by-line, updating values as the algorithm dictates.
- Spots infinite loops, off-by-one errors, missing updates, wrong conditions.
- When flowchart shows wrong logic, write a corrected pseudo-code version.

Crib sheet »

Algorithms — Sorts, Searches, Recursion & Big-O Complexity in one page

Quick-reference notes — revisit before each question.

Pseudocode template

Declare variables and types.
FOR / WHILE / REPEAT loops indented.
IF / THEN / ELSE for selection.
Use clear identifiers and end-markers (end while, next i).

Bubble sort

```
FOR i = 0 to n-2
 FOR j = 0 to n-i-2
  IF a[j] > a[j+1] THEN swap a[j], a[j+1]
```

$O(n^2)$ worst, $O(n)$ best (with swap flag).

Binary search

```
first ← 0; last ← n-1
REPEAT
 m ← (first + last) DIV 2
 IF target < a[m] THEN last ← m-1 ELSE
first ← m+1
UNTIL a[m] = target
```

$O(\log n)$

Quicksort

Pick pivot.
Partition: everything ≤ pivot to its left, everything > pivot to its right.
Recurse on each side.
Average $O(n \log n)$; worst $O(n^2)$ on already-sorted input.

Big-O quick map

Single loop ⇒ $O(n)$.
Loop halving range ⇒ $O(\log n)$.
Nested loop ⇒ $O(n^2)$.
Triple-nested loop ⇒ $O(n^3)$.
Const work, no loop ⇒ $O(1)$.

Dry-run discipline

One row per loop iteration.
Column per variable + output column.
Tick condition outcomes as you go.
Stop on the cycle that fails — that's the bug.

5. *(a)* Using the laws of Boolean algebra and De Morgan's theorem simplify the following Boolean expression:

$$A \cdot (\overline{A \cdot B})$$ [4]

(b) Simplify the following Boolean expression:

$$A \cdot B + A \cdot (B + C) + B \cdot (B + C)$$ [4]

6. Write a bubble sort algorithm, using pseudo-code, to sort an array of integers into ascending order. [9]

7. The name of a constant in a certain computer language must either be a single uppercase letter, or a single uppercase letter followed by one or more uppercase letters or digits.

(a) Produce an appropriate syntax diagram to define a constant in this language. [4]

(b) Produce an appropriate Backus-Naur Form (BNF) definition for a constant in this language. [4]

8. Below is part of an algorithm that initialises a two dimensional array named GRID of size N x N with zeros.

```
for i = 0 to N - 1
          for j =  0 to N - 1
                GRID(i,j) = 0
          next j
    next i
```

Evaluate the efficiency of the algorithm and, using Big O notation, determine the growth rate for the time performance. [4]

9. *(a)* Define the term algorithm. Other than pseudo-code, state **one** method of expressing an algorithm. [2]

(b) Describe the main characteristics of a recursive algorithm. [2]

(c) Describe **two** disadvantages of using a recursive algorithm. [2]

5. *(a)* Using the laws of Boolean algebra and De Morgan's theorem simplify the following Boolean expression:

$$A \cdot (\overline{A \cdot B})$$ [4]

(b) Simplify the following Boolean expression:

$$A \cdot B + A \cdot (B + C) + B \cdot (B + C)$$ [4]

6. Write a bubble sort algorithm, using pseudo-code, to sort an array of integers into ascending order. [9]

7. The name of a constant in a certain computer language must either be a single uppercase letter, or a single uppercase letter followed by one or more uppercase letters or digits.

(a) Produce an appropriate syntax diagram to define a constant in this language. [4]

(b) Produce an appropriate Backus-Naur Form (BNF) definition for a constant in this language. [4]

8. Below is part of an algorithm that initialises a two dimensional array named GRID of size N x N with zeros.

```
for i = 0 to N - 1
          for j =  0 to N - 1
                GRID(i,j) = 0
          next j
    next i
```

Evaluate the efficiency of the algorithm and, using Big O notation, determine the growth rate for the time performance. [4]

9. *(a)* Define the term algorithm. Other than pseudo-code, state **one** method of expressing an algorithm. [2]

(b) Describe the main characteristics of a recursive algorithm. [2]

(c) Describe **two** disadvantages of using a recursive algorithm. [2]

5. *(a)* Using the laws of Boolean algebra and De Morgan's theorem simplify the following Boolean expression:

$$A \cdot (\overline{A \cdot B})$$ [4]

(b) Simplify the following Boolean expression:

$$A \cdot B + A \cdot (B + C) + B \cdot (B + C)$$ [4]

6. Write a bubble sort algorithm, using pseudo-code, to sort an array of integers into ascending order. [9]

7. The name of a constant in a certain computer language must either be a single uppercase letter, or a single uppercase letter followed by one or more uppercase letters or digits.

(a) Produce an appropriate syntax diagram to define a constant in this language. [4]

(b) Produce an appropriate Backus-Naur Form (BNF) definition for a constant in this language. [4]

8. Below is part of an algorithm that initialises a two dimensional array named GRID of size N x N with zeros.

```
for i = 0 to N - 1
          for j =  0 to N - 1
                GRID(i,j) = 0
          next j
    next i
```

Evaluate the efficiency of the algorithm and, using Big O notation, determine the growth rate for the time performance. [4]

9. *(a)* Define the term algorithm. Other than pseudo-code, state **one** method of expressing an algorithm. [2]

(b) Describe the main characteristics of a recursive algorithm. [2]

(c) Describe **two** disadvantages of using a recursive algorithm. [2]

6. An algorithm is shown below.

```
1    decNumber is integer
2    bin is integer
3    answer is string
4
5    while decNumber <> 0
6          if decNumber MOD 2 = 0 then
7                bin = 0
8          else
9                bin = 1
10         end if
11         answer = char(bin) + answer
12         decNumber = decNumber DIV 2
13   end while
14
15   output answer
16
17   end subroutine
```

(a) Dry run the algorithm using the trace table below. [2]

decNumber	decNumber MOD 2	bin	answer
137			

(b) State the purpose of the algorithm. [1]

(c) Explain the selection of data types for “bin” and “answer” in the algorithm. [3]

7. An Institute for ICT technicians in schools and colleges operates a code of conduct.

(a) Describe the purpose of the code of conduct. [2]

(b) Identify **two** standards that should be included in the code for professional competence. [2]

(c) Identify **two** standards that should be included in the code for professional integrity. [2]

8. An online retailer offers a large range of stock items. They use a hash table to store details of the stock items in their computer based stock control system. Each stock item has a key value.

(a) Explain the operation of a hash table and why the time taken to perform search and insertion operations is not affected by the number of stock items stored. [3]

(b) The retailer needs to store customers' delivery details, including their postal address. In this system a postal address is made up of the street, town and postcode.

Street can be a house number or name followed by the name of the street.
All towns begin with an uppercase letter followed by lowercase letters.
All letters in a postcode will be uppercase.
Postcodes can include two or three digits.

Produce an appropriate Backus-Naur Form (BNF) definition for this postal address. [8]

9. The two sections of code below carry out the same task. One section of code uses iteration and the other recursion.

```
Declare subFactorial
fact = 1
input n
for i = 1 to n
     fact = fact * i
next i
print "factorial of", n "is", fact
end sub
```

```
function factorial(n) is integer
if n <= 1 then
     return 1
else
     return factorial = n * factorial(n-1)
end if
end function
```

(a) Describe iteration. [2]

(b) Describe recursion. [2]

(c) Describe **two** advantages of using an iterative function compared with a recursive function to solve a given problem. [4]

10. *(a)* Describe the purpose of data validation. [2]

(b) Write an algorithm to validate a date in dd/mm/yyyy format. [11]

11. *(a)* Describe what is meant by Object-Oriented Programming (OOP). [4]

(b) Explain the relationship between classes and instances. [3]

(c) Explain what is meant by a method. [3]

12. Explain the need for standardisation of computer languages and discuss the advantages arising from the use of algorithms and programming languages that have been standardised.

You should draw on your knowledge, skills and understanding from a number of areas across your Computer Science course when answering this question. [10]

END OF PAPER

5. Below is an algorithm.

```
Declare subprocedure myAlgorithm(myArray is integer, indexLow is
integer, indexHi is integer)

Declare pivot is integer
Declare tmpSwap is integer
Declare tmpLow is integer
Declare tmpHi is integer

set tmpLow = indexLow
set tmpHi = indexHi

set pivot = myArray[(int((indexLow + indexHi)/2))]

while (tmpLow <= tmpHi)

     while (myArray[tmpLow] < pivot and tmpLow < indexHi)
           set tmpLow = tmpLow + 1
     end while

     while (pivot < myArray[tmpHi] and tmpHi > indexLow)
           set tmpHi = tmpHi - 1
     end while

     if (tmpLow <= tmpHi) then
           set tmpSwap = myArray[tmpLow]
           set myArray[tmpLow] = myArray[tmpHi]
           set myArray[tmpHi] = tmpSwap
           set tmpLow = tmpLow + 1
           set tmpHi = tmpHi - 1
     end if
end while

if (indexLow < tmpHi) then myAlgorithm(myArray, indexLow, tmpHi)
if (tmpLow < indexHi) then myAlgorithm(myArray, tmpLow, indexHi)
```

(a) Describe the purpose of this algorithm. [2]

(b) Describe the characteristics of this type of algorithm. [3]

(c) Describe the advantages arising from the elegance of this algorithm. [3]

9. Below are two algorithms that search for a data item in a one dimensional array. You can assume that the data in the array is in ascending order and that the data item being searched is present. The Search_A algorithm has a time performance of O(n).

```
Algorithm Search_A

declare searchKey, i as integer
declare flag as Boolean
declare myArray[] as integer[]
Input searchKey
Set i = 0
flag = FALSE

repeat
      if myArray[i] = searchKey then
            set flag = TRUE
      end if
      set i = i + 1
until (flag = TRUE)
output myArray[i]
```

```
Algorithm Search_B

declare searchKey, first, last, m as integer
declare myArray[] as integer[]
Input searchKey
Set first = 1
Set last = len(myArray[])

repeat
      set m = (first + last) DIV 2
      if searchKey < myArray[m] then
            set last = m - 1
      else
            set first = m + 1
      end if
until (myArray[m] = searchKey)
output myArray[m]
```

(a) Evaluate the efficiency of the Search_B algorithm and using Big O notation, determine the growth rate for time performance. [5]

(b) Draw a graph of the algorithms above to illustrate their order of time performance. Graph paper is not required. [4]

(c) State which algorithm is more efficient when searching for a data item. [1]

10. Describe the term data compression and explain how data compression algorithms are used.[6]

11. *(a)* Describe what is meant by a class and an object, and describe the relationship between them. [4]

(b) Describe what is meant by the term method, and describe the relationship between object and method. [2]

9. Below are two algorithms that search for a data item in a one dimensional array. You can assume that the data in the array is in ascending order and that the data item being searched is present. The Search_A algorithm has a time performance of O(n).

```
Algorithm Search_A

declare searchKey, i as integer
declare flag as Boolean
declare myArray[] as integer[]
Input searchKey
Set i = 0
flag = FALSE

repeat
      if myArray[i] = searchKey then
            set flag = TRUE
      end if
      set i = i + 1
until (flag = TRUE)
output myArray[i]
```

```
Algorithm Search_B

declare searchKey, first, last, m as integer
declare myArray[] as integer[]
Input searchKey
Set first = 1
Set last = len(myArray[])

repeat
      set m = (first + last) DIV 2
      if searchKey < myArray[m] then
            set last = m - 1
      else
            set first = m + 1
      end if
until (myArray[m] = searchKey)
output myArray[m]
```

(a) Evaluate the efficiency of the Search_B algorithm and using Big O notation, determine the growth rate for time performance. [5]

(b) Draw a graph of the algorithms above to illustrate their order of time performance. Graph paper is not required. [4]

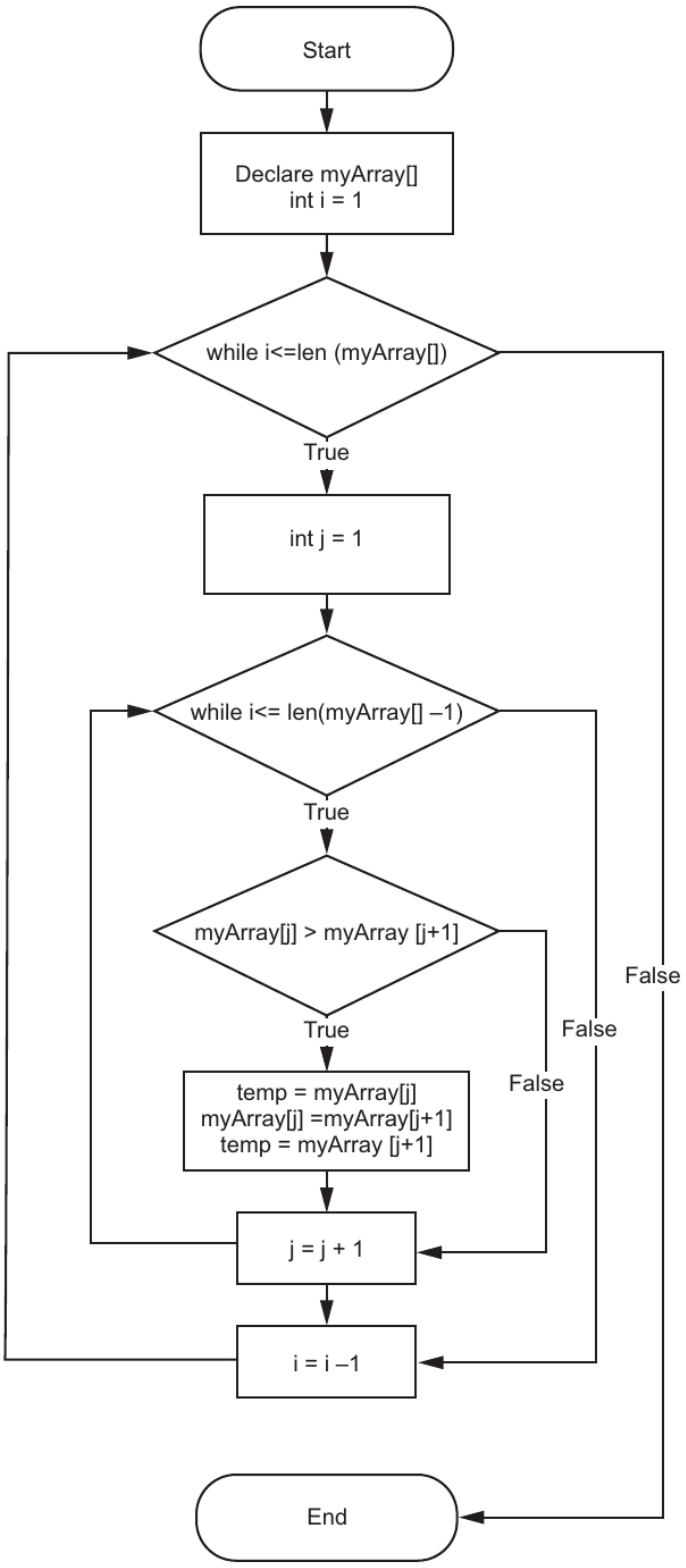
(c) State which algorithm is more efficient when searching for a data item. [1]

10. Describe the term data compression and explain how data compression algorithms are used.[6]

11. *(a)* Describe what is meant by a class and an object, and describe the relationship between them. [4]

(b) Describe what is meant by the term method, and describe the relationship between object and method. [2]

12. The algorithm in the flowchart is intended to sort an array in descending order. However, the algorithm contains several errors. Write a corrected version of the algorithm in pseudo-code.
[7]



Answer **all** questions.

1. (a) Explain, giving a suitable example, the operation of a stack data structure. [4]

(b) This is a graphical representation of a two-dimensional array:

alphabet[ ]

	0	1	2	3	4	5	6
0	'a'	'b'	'c'	'd'	'e'	'f'	'g'
1	'h'	'i'	'j'	'k'	'l'	'm'	'n'
2	'o'	'p'	'q'	'r'	's'	't'	'u'
3	'v'	'w'	'x'	'y'	'z'	' '	'.'

(i) Describe how you would update the value **'.'** to **'!'** in the alphabet array. [2]

(ii) Demonstrate how a two-dimensional array could be used to store pupil data in school. [2]

2. (a) Define the term algorithm. [2]

(b) Describe one method of defining algorithms. [2]

(c) Explain the benefits and drawbacks of recursive and non-recursive algorithms. [4]

3. Clearly showing each step, simplify the following Boolean expressions using Boolean algebra, identities and De Morgan's Law where appropriate.

(a) $(A + C).(A.C + A.\overline{C}) + A.C$ [6]

(b) $\overline{X.Y}\ .(\overline{X} + Y) + \overline{Z}$ [5]

4. Write a Binary Search algorithm to search for a value in an array of characters and output its position. You may assume that the search value exists in the array. [8]

Answer **all** questions.

1. (a) Explain, giving a suitable example, the operation of a stack data structure. [4]

(b) This is a graphical representation of a two-dimensional array:

alphabet[ ]

	0	1	2	3	4	5	6
0	'a'	'b'	'c'	'd'	'e'	'f'	'g'
1	'h'	'i'	'j'	'k'	'l'	'm'	'n'
2	'o'	'p'	'q'	'r'	's'	't'	'u'
3	'v'	'w'	'x'	'y'	'z'	' '	'.'

(i) Describe how you would update the value **'.'** to **'!'** in the alphabet array. [2]

(ii) Demonstrate how a two-dimensional array could be used to store pupil data in school. [2]

2. (a) Define the term algorithm. [2]

(b) Describe one method of defining algorithms. [2]

(c) Explain the benefits and drawbacks of recursive and non-recursive algorithms. [4]

3. Clearly showing each step, simplify the following Boolean expressions using Boolean algebra, identities and De Morgan's Law where appropriate.

(a) $(A + C).(A.C + A.\overline{C}) + A.C$ [6]

(b) $\overline{X.Y}\ .(\overline{X} + Y) + \overline{Z}$ [5]

4. Write a Binary Search algorithm to search for a value in an array of characters and output its position. You may assume that the search value exists in the array. [8]

5. The following is an algorithm.

```
1  declare subprocedure DivMod
2
3  x, y, a, b, c, ds is integer
4
5  output "Enter a three-digit integer (100 – 999): "
6  input x
7  if x > 99 AND x < 1000 then
8      a = x DIV 100
9      b = x MOD 100
10     c = x MOD 10
11     b = (b – c) DIV 10
12     output a
13     output b
14     output c
15     ds = (a + b + c)
16     output ds
17     y = ((a * 100) + (b * 10) + c)
18     output y
19 else
20      output "error"
21 end if
22
23 end procedure
```


(a) Select appropriate test data to dry run the algorithm and give all outputs. [6]

(b) Explain the purpose of the algorithm. [2]

11. This algorithm duplicates a three-dimensional array of length *n*. You can assume the array (myArray) has already been populated with data.

```
1    declare i,j,k,n as integer
2    declare myArray[n,n,n]
3    declare myArrayCopy[n,n,n]
4
5    set i = 0
6    set j = 0
7    set k = 0
8
9    for i = 0 to n - 1
10
11         for j = 0 to n - 1
12
13                for k = 0 to n - 1
14
15                      set myArrayCopy[i,j,k] = myArray[i,j,k]
16
17                next k
18
19         next j
20
21   next i
```

(a) Evaluate the efficiency of the algorithm and using Big O notation, determine the growth rate for time performance. [5]

(b) Determine the growth rate of memory space during a single run of the algorithm. [2]

(c) Identify the type of time complexity and draw a graph of the algorithm to illustrate the order of time performance. Graph paper is not required. [2]

12. Describe how data may be recovered if lost. [4]

13. Discuss the importance of codes of conduct in promoting professional behaviour throughout the software development stages.

You should draw on your knowledge, skills and understanding from a number of areas across your computer science course when answering this question. [12]

END OF PAPER

4. Clearly showing each step, simplify the following Boolean expressions using Boolean algebra, identities and De Morgan's Law where appropriate.

(a) $A.(1+C)+\overline{B}.(A+B)$ [5]

(b) $X.\left(\overline{Y+Z}\right)+\overline{Z}.X$ [5]

5. Write a quicksort algorithm, to sort an array of integers into ascending order. [8]

6. State the purpose of validation and verification, giving a suitable method for each. [4]

7. An online grocery store uses binary trees. These binary trees can be traversed using a variety of methods.

(a) Describe the following methods of traversal and give an example of why each method would be used in the grocery store.

(i) In-order traversal [3]

(ii) Post-order traversal [3]

(iii) Pre-order traversal [3]

(b) Draw an example of a balanced binary tree for grocery items. [1]

8. Giving suitable examples, describe the types of software tool that have been designed to assist in the following:

(a) Analysis and planning [3]

(b) Software development [3]

(c) Version management [3]

12. This is an algorithm which searches for consecutive data items in three separate one dimensional arrays all of size *n*. You can assume all the arrays have already been populated with data.

```
Algorithm Search

declare i as integer
declare myArray1[n] as integer[]
declare myArray2[n] as integer[]
declare myArray3[n] as integer[]
declare found as string

set i = 0
set found = ""
do

     if myArray1[i] = myArray1[i+1] then
          set found = found + myArray1[i] + " "
     end if

     if myArray2[i] = myArray2[i+1] then
          set found = found + myArray2[i] + " "
     end if

     if myArray3[i] = myArray3[i+1] then
          set found = found + myArray3[i] + " "
     end if

     set i = i + 1

while (i < n)
output "Consecutive data items found: " , found
```

(a) Evaluate the efficiency of the algorithm and using Big O notation, determine the growth rate for time performance. [5]

(b) Draw a graph of the algorithm to illustrate and identify its order of time performance. Graph paper is not required. [4]

13. Discuss contemporary approaches to human-computer interaction.

You should draw on your knowledge, skills and understanding from a number of areas across your computer science course when answering this question. [13]

END OF PAPER

2. Computer languages such as HTML and CSS are standardised.

(a) Describe the need for the standardisation of computer languages. [4]

(b) Explain, giving a suitable example, the potential difficulties involved in agreeing and implementing standards. [4]

3. Clearly showing each step, simplify the following Boolean expressions using Boolean algebra, identities and De Morgan's Law where appropriate.

(a) $(1.X + \overline{Y.Y}).\overline{X} + X.Z$ [6]

(b) $(\bar{A}.A + A.A).1 + B.C.0$ [6]

4. Write an algorithm to search an unsorted one-dimensional array of strings and replace any duplicate values with the string "X" and then output the array. [8]

5. This algorithm validates a password for a given string.

```
1    declare password as string
2    declare digits as char[]
3    declare special as char[]
4    declare length, i as integer
5    declare valid, w, x, y, z as Boolean
6    set digits = {'0','1','2','3','4','5','6','7','8','9'}
7    set special = {'!','@','£','*','?'}
8    set length = 8
9    set i, j = 0
10   set valid, x, y, z = FALSE
11
12   do
13       output "Enter a password: "
14       input password
15       if len(password) >= length
16            set x = TRUE
17       end if
18       for i = 0 to len(password) - 1
19            for j = 0 to len(digits) - 1
20                 if password[i] = digits[j]
21                      set y = TRUE
22                 end if
23            next j
24       next i
25       set j = 0
26       for i = 0 to len(password) - 1
27            for j = 0 to len(special) - 1
28                 if password[i] = special[j]
29                      set z = TRUE
30                 end if
31            next j
32       next i
33       if NOT x OR NOT y OR NOT z then
34            output "message 1"
35       else
36            output "message 2"
37            set valid = TRUE
38   while valid = FALSE
```

(a) Using a table, select **three** appropriate pieces of test data to dry-run the algorithm and trace the variables `x`, `y`, `z`, `valid` and the expected output. [6]

(b) Suggest suitable text for `message 1` and `message 2`. [2]

6. When evaluating computer-based solutions, there are several criteria that can be considered.

Describe the following criteria when evaluating computer-based solutions and give suitable examples:

(a) Usability [2]

(b) Performance [2]

(c) Scalability [2]

(d) Security [2]

7. A website URL is made up of a protocol, a domain name, and an optional file path.

- The protocol can only be “http” or “https”.
- The domain name can only consist of alphanumeric characters, hyphens and full stops.
- The protocol and domain name must be separated by a colon and two forward slashes.
- The optional file path must start with a forward slash and can only contain alphanumeric characters and forward slashes.

Example: https://www.wjec.co.uk/home/

Produce a Backus-Naur form (BNF) definition for a valid website URL. [6]

8. Explain program version management. [6]

9. Explain, using suitable examples, recursive and non-recursive sorting algorithms. [8]

10. This is a signed 8-bit integer:

$$00001111_2$$

Include this integer in a worked example to demonstrate how masking can be used to determine the sign of the integer. [3]

11. This algorithm merges two one-dimensional arrays of length *n*. Assume the arrays (`myArray1` and `myArray2`) have already been populated with data.

`num(array)` returns the number of elements currently in the array.

```
1    declare i,j as integer
2    declare myArray1[n]
3    declare myArray2[n]
4    declare myArray3[]
5
6    set i = 0
7    set j = 0
8
9    for i = 0 to len(myArray1) - 1
10        set myArray3[num(myArray3)] = myArray1[i]
11   next i
12
13   for j = 0 to len(myArray2) - 1
14        set myArray3[num(myArray3)] = myArray2[j]
15   next j
16
17   output myArray3
```

(a) Evaluate the efficiency of the algorithm and, using Big O notation, determine the growth rate for time performance. [5]

(b) Determine the growth rate of memory space during a single run of the algorithm. [2]

(c) Identify the type of time complexity and draw a graph of the algorithm to illustrate the order of time performance. Graph paper is not required. [2]

12. Explain, giving a suitable example, the shortest path algorithm. [4]

13. Discuss the potential use of natural language interfaces in human-computer communication to address the problem of communicating with computers.

You should draw on your knowledge, skills and understanding from a number of areas across your computer science course when answering this question. [12]

END OF PAPER

END OF QUESTION PACK

20 questions · 160 marks · ~4 h
Source: WJEC A2 Computer Science Unit 3 (1500U30-1), Summer 2017–2024, COVID gap
Curated for WJEC Computer Science 2015 spec A2 Unit 3 — Topic 3 (3.3)