Past Papers May/June 2015 to 2024:

9618/22/M/J/24

1 (a) The following table contains pseudocode examples
Each example may contain statements that relate to one or more of the following:

- Selection
- iteration (repetition)
- input/output.

Complete the table by placing **one or more ticks** ( ) in each row.

Pseudocode Example	Selection	Iteration	Input/Output

`FOR Index 1 TO 10`
`Data[Index] 0`
`NEXT Index`			

`WRITEFILE ThisFile, "****"`			
`UNTIL Level > 25`			

`IF Mark > 74 THEN`
`READFILE OldFile, Data`
`ENDIF`			

[4]

(b) Program variables have data types as follows:

Variable	Data type
MyChar	CHAR
MyString	STRING
MyInt	INTEGER

Complete the table by filling in each gap with a function (from the **insert**) so that each expression is valid.

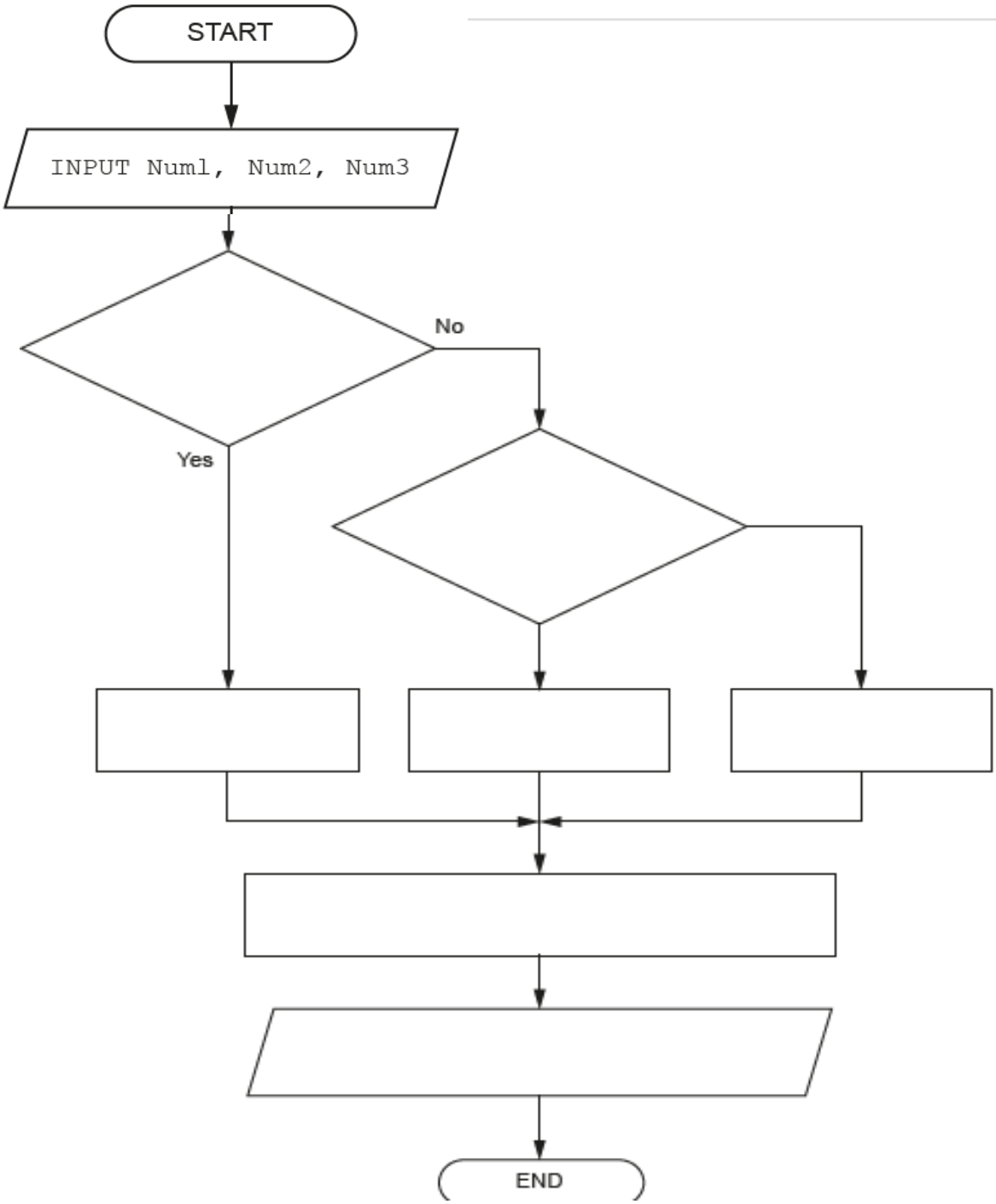
Expression
MyInt ⟵ ..... (3.1415926)
MyChar ⟵ ..... ("Elwood", 3, 1)
MyString ⟵ ..... (27.509))
MyInt ⟵ ..... (..... ("ABC123", 3))

[4]

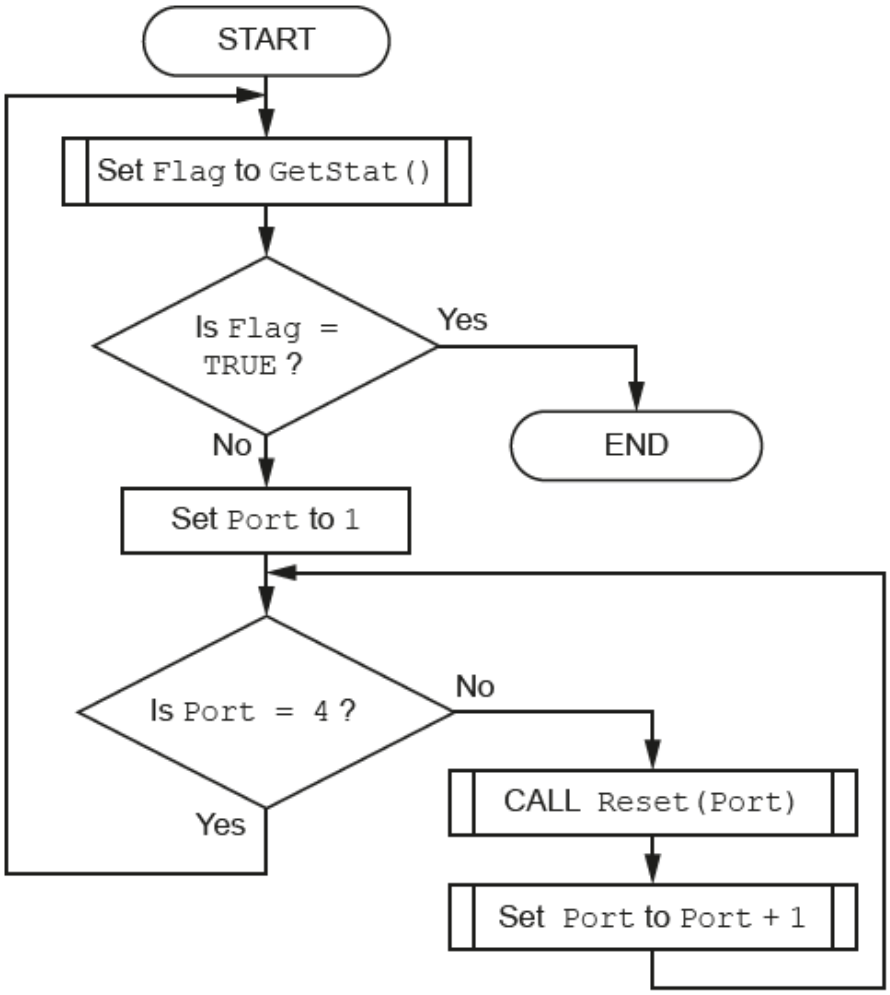
2 A program is being developed.
(a) An algorithm for part of the program will:

- input three numeric values and assign them to identifiers **`Num1, Num2 and Num3`**
- assign the largest value to variable **`Ans`**
- output a message giving the **largest value** and the **average** of the three numeric values.

Assume the values are all different and are input in no particular order.
Complete the program flowchart on page 5 to represent the algorithm.



(b) A different part of the program contains an algorithm represented by the following program flowchart:



Write pseudocode for the algorithm.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

3. (a) A 1D array **`Item`** of **2000 elements** will store the data for all items.
Write pseudocode to declare the `Item` array.

.....

..... [2]

(b) State **three** benefits of using an array of records to store the data for all items.

1 .....

.....

2 .....

.....

3 .....

.....[3]

9608/21/22/MJ/16

4 (a) Structured programming involves the breaking down of a problem into modules.
Give two reasons why this is done.

1 .....

.....

.....

.....

2 .....

.....

.....

.....

..... [2]

9608/22/MJ/18

5 A chocolate factory produces bars of chocolate. A computer program controls the process.

The weight of each bar is stored in an array, **`BarWeight.`** The array contains 100 elements, representing the weights of 100 bars that make up one shipping box.

A procedure, **`CheckWeight()`**, is required to:

1. examine each array element and count how many times the weight has exceeded **`MaxWeight`**
2. compare the count obtained with a limit value, **`Threshold`**. Call procedure **`ServiceCheck()`** if the count exceeds the **`Threshold`**
3. output a message if the count does not exceed the **`Threshold`**. For example: "ShippingBox OK – maximum weight exceeded 3 times."

Draw a program flowchart on the next page to represent the algorithm for the **`CheckWeight()`** procedure.

Assume that:

- the array contains 100 valid weight values and the first element is **`BarWeight`**[1]
- **`MaxWeight`**, **`Threshold`** and **`BarWeight`** are global variables. Variable declarations are not required in program flowcharts.

(9608/21/ON/15)

Q6. A firm employs five staff who take part in a training program. Each member of staff must complete a set of twelve tasks which can be taken in any order. When a member of staff successfully completes a task, this is recorded.

A program is to be produced to record the completion of tasks for the five members of staff. To test the code, the programmer makes the program generate test data.

The program generates pairs of random numbers:

- the first, in the range, 1 to 5 to represent the member of staff
- the second, in the range, 1 to 12 to represent the task

Each pair of numbers simulates the completion of one task by one member of staff.

(a) Explain why the generation of 60 (5 staff x 12 tasks) pairs of random numbers will not simulate all tasks completed by all staff.

.....

.....

.....

.....

.....[2]

(b) Data is currently recorded manually as shown.

Staff number	Task number											
	1	2	3	4	5	6	7	8	9	10	11	12
1												
2												
3				✓								
4												
5								✓				

The table shows that two members of staff have each successfully completed one task.
The program must use a suitable data structure to store, for all staff:

- tasks successfully completed
- tasks not yet successfully completed

The program will output the staff number and task number in the order in which tasks are completed.

The program design in pseudocode is produced as follows:

```
01 DECLARE StaffNum : INTEGER
02 DECLARE TaskNum : INTEGER
03 DECLARE.....................................................
04 DECLARE NewStaffTask : BOOLEAN
05
06 CALL InitialiseTaskGrid
07 Completed ← 0
08 WHILE Completed <> 60
09   NewStaffTask ← FALSE
10   WHILE NewStaffTask = FALSE
11      StaffNum ← RANDOM(1,5)    //generates a random number
12      TaskNum ← RANDOM(1,12)    //in the given range
13      IF TaskGrid[StaffNum, TaskNum] = FALSE
14         THEN
15            TaskGrid[StaffNum, TaskNum] ← TRUE
16            NewStaffTask ← TRUE
17            OUTPUT StaffNum, TaskNum
18      ENDIF
19   ENDWHILE
20   Completed ← Completed + 1
21 ENDWHILE
22 OUTPUT "Staff Task Count", Completed
23
24 // end of main program
25
26 PROCEDURE InitialiseTaskGrid()
27   DECLARE i : INTEGER
28   DECLARE j : INTEGER
29   FOR i ← 1 TO 5
30      FOR j ← 1 TO 12
31         TaskGrid[i, j] ← FALSE
32      ENDFOR
33   ENDFOR
34 ENDPROCEDURE
```

Study the pseudocode and answer the questions below.

Give the line number for:

(i) The declaration of a BOOLEAN global variable. ..... [1]

(ii) The declaration of a local variable. ..... [1]

(iii) The incrementing of a variable used as a counter, but not to control a 'count controlled' loop. ..... [1]

(iv) A statement which uses a built-in function of the programming language..... [1]

(c)

(i) State the number of parameters of the InitialiseTaskGrid procedure. ..... [1]

(ii) Copy the condition which is used to control a 'pre-condition' loop.

.....[1]

(iii) Explain the purpose of lines 13 – 18.

.....

.....

.....

.....

.....

.....[3]

(iv) Give the global variable that needs to be declared at line 03.

.....[2]

7 (a) (i) Complete the following table by giving the appropriate data type in each case.

Variable	Example data value	Data type
Name	"Catherine"	
Index	100	
Modified	FALSE	
Holiday	25/12/2020	

[4]

(ii) Evaluate each expression in the following table by using the initial data values shown in **part (a)(i)**.

Expression	Evaluates to
`Modified OR Index > 100`	
`LENGTH("Student: " & Name)`	
`INT(Index + 2.9)`	
`MID(Name, 1, 3)`	

[4]

(b) Each pseudocode statement in the following table contains an example of selection, assignment or iteration.
Put **one** tick ('✓') in the appropriate column for each statement.

Statement	Selection	Assignment	Iteration
`Index ← Index + 1`			
`IF Modified = TRUE THEN`			
`ENDWHILE`			

[3]

Built-in functions

STRING Functions

LEFT(ThisString : STRING, x : INTEGER) RETURNS STRING returns
leftmost x characters from ThisString
Example: LEFT("ABCDEFGH", 3) returns "ABC"

RIGHT(ThisString : STRING, x : INTEGER) RETURNS STRING returns
rightmost x characters from ThisString
Example: RIGHT("ABCDEFGH", 3) returns "FGH"

MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING returns a
string of length y starting at position x from ThisString
Example: MID("ABCDEFGH", 2, 3) returns "BCD"

LENGTH(ThisString : STRING) RETURNS INTEGER returns the integer value representing the length of ThisString
Example: LENGTH("Happy Days") returns 10

LCASE(ThisChar : CHAR) RETURNS CHAR returns the character value representing the lower case equivalent of ThisChar Characters that are not upper case alphabetic are returned unchanged
Example: LCASE('W') returns 'w'

UCASE(ThisChar : CHAR) RETURNS CHAR
returns the character value representing the upper case equivalent of ThisChar
Characters that are not lower case alphabetic are returned unchanged
Example: UCASE('a') returns 'A'

TO_UPPER(ThisString : STRING) RETURNS STRING
returns a string formed by converting all characters of ThisString to upper case

Example: TO_UPPER("Error 803") returns "ERROR 803"

TO_LOWER(ThisString : STRING) RETURNS STRING
returns a string formed by converting all characters of ThisString to lower case
Example: TO_LOWER("JIM 803") returns "jim 803"

NUM_TO_STR(x : <data type>) RETURNS STRING
returns a string representation of a numeric value
Note: <data type> may be REAL or INTEGER

Example: NUM_TO_STR(87.5) returns "87.5"

STR_TO_NUM(x : <data type1>) RETURNS <data type2>
returns a numeric representation of a string Note: <data
type1> may be CHAR or STRING
Note: <data type2> may be REAL or INTEGER
Example: STR_TO_NUM("23.45") returns 23.45

IS_NUM(ThisString : STRING) RETURNS BOOLEAN

returns the value TRUE if ThisString represents a valid numeric
value Note: <data type> may be CHAR or STRING

Example: IS_NUM("12.36") returns TRUE

Example: IS_NUM("-12.36") returns TRUE

Example: IS_NUM("12.3a") returns FALSE

ASC(ThisChar : CHAR) RETURNS INTEGER returns
an integer value (the ASCII value) of ThisChar
Example: ASC('A') returns 65

CHR(x : INTEGER) RETURNS CHAR

returns the character whose integer value (the ASCII

value) is x Example: CHR(87)returns 'W'

NUMERIC Functions

INT(x : REAL) RETURNS INTEGER

returns the integer part of x

Example: INT(27.5415) returns 27

RAND(x : INTEGER) RETURNS REAL

returns a real number in the range 0 to x (**not** inclusive of x)

Example: RAND(87) could return 35.43

DATE Functions

Note: Date format is assumed to be DDMMYYYY unless otherwise stated.

DAY(ThisDate : DATE) RETURNS INTEGER

returns the current day number from ThisDate

Example: DAY(4/10/2003) returns 4

MONTH(ThisDate : DATE) RETURNS
INTEGER returns the current month number
from ThisDate
Example: MONTH(4/10/2003) returns 10

YEAR(ThisDate : DATE) RETURNS INTEGER

returns the current year number from ThisDate

Example: YEAR(4/10/2003) returns 2003

DAYINDEX(ThisDate : DATE) RETURNS INTEGER

returns the current day index number from ThisDate where Sunday = 1, Monday = 2, Tuesday = 3 etc. Example: DAYINDEX(12/05/2020) returns 3

SETDATE(Day, Month, Year : INTEGER) RETURNS DATE

returns a variable of type DATE

NOW() RETURNS DATE

returns the current date

MARKING KEY

9618/22/M/J/24

1(a)

Pseudocode example	Selection	Iteration	Input/Output

`FOR Index ← 1 TO 10`
`  Data[Index] ← 0`
`NEXT Index`
		✓	

`WRITEFILE ThisFile, "****"`			✓
`UNTIL Level > 25`		✓	

`IF Mark > 74 THEN`
`  READFILE OldFile, Data`
`ENDIF`
	✓		✓

One mark per row. [4]

1(b)

Expression
`MyInt ← INT (3.1415926)`
`MyChar ← MID ("Elwood", 3, 1)`

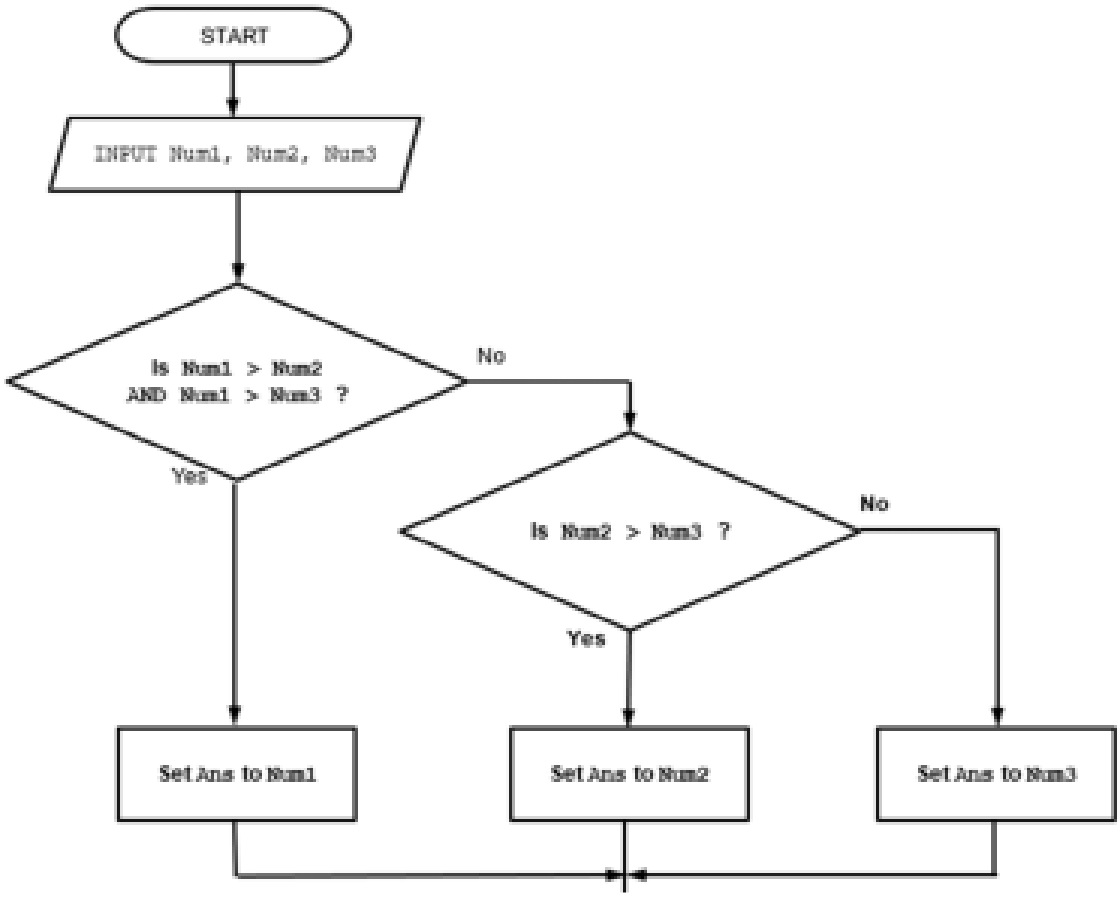
Any of:

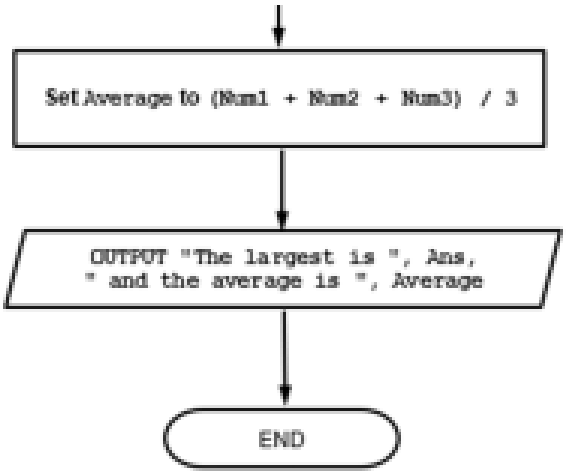
```
MyString = NUM_TO_STR (INT (27.509))
MyString = CHR (INT (27.509))
MyString = TO_UPPER( NUM_TO_STR( 27.509))
MyString = TO_LOWER( NUM_TO_STR( 27.509))

Any of:
MyInt = STR_TO_NUM ( RIGHT ("ABC123", 3))
MyInt = LENGTH ( RIGHT ("ABC123", 3))
MyInt = LENGTH ( LEFT ("ABC123", 3))
```

[4]

2(a)





Mark points
1 Condition for selecting one of the input numbers as largest value
2 … and assign to Ans
3 Condition for selecting largest number for all three number input and assigning to Ans
4 Average calculated using Num1, Num2 and Num3 and stored in a variable.
Reject use of DIV
5 Output of Ans and average in output symbol / parallelogram

[5]

2(b) Example solutions:

```
Flag = GetStat()
WHILE Flag <> TRUE
      FOR Port = 1 TO 3
            CALL Reset(Port)
      NEXT Port
      Flag = GetStat()
ENDWHILE
```

```
Alternative:
REPEAT
      Flag = GetStat()
            IF Flag <> TRUE THEN
                  FOR Port = 1 TO 3
                        CALL Reset(Port)
                  NEXT Port
            ENDIF
UNTIL Flag = TRUE
```

[5]

One mark per point:
1 (Outer) conditional loop testing Flag

2 Correct assignment of Flag from GetStat() **in a loop**
3 (Inner) loop checking / counting port // Check if Port is different to 4
4 … loop for 3 iterations
5 a call to Reset()**in a loop**

3(a)(ii)

```
DECLARE Item : ARRAY [1:2000] OF Component//
DECLARE Item : ARRAY [2000] OF Component//
DECLARE Item : ARRAY [0:1999] OF Component
```

One mark per underlined phrase **[2]**

3(b) One mark per point:
1 Allows for iteration / can use a loop **to access the records / data items**
2 Use of index to directly access a **record** in the array // Example of simplification of code e.g. use of dot notation Item[1].Stage
3 Simplifies the code / algorithm // Reduces duplication of code // **Program** easier to write / understand / maintain / test / debug // **Data items/record** easier to search / sort / manipulate

[3]

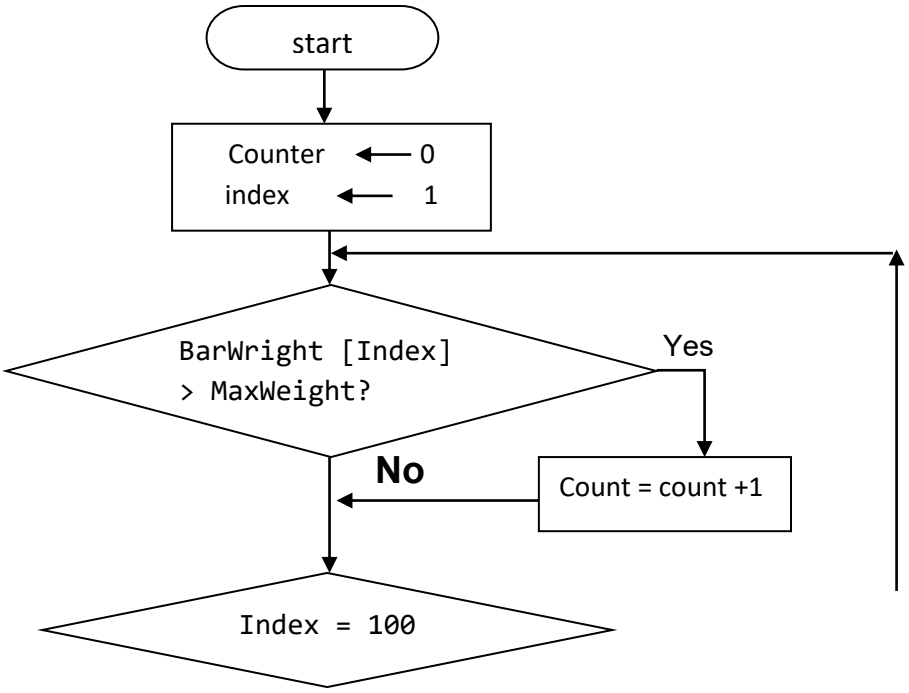
9608/21/22/MJ/16
Answer
4 (a)

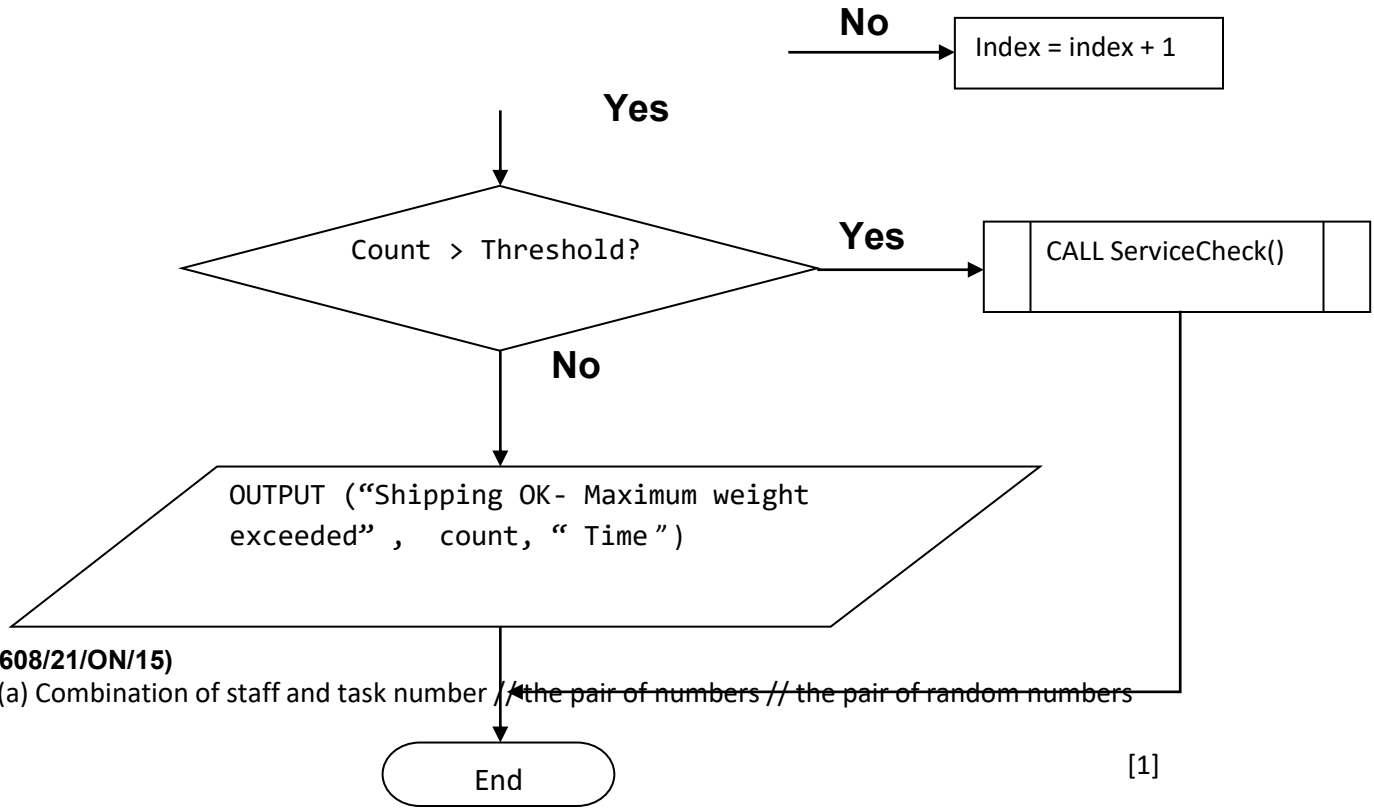
- Program code is easier to implement / manage
- Modules may be given to different people to develop // given to program specialists
- Program code is easier to test / debug / maintain
- Encourages the re-usability of program code

[2]

9608/22/MJ/18
5

[10]





(9608/21/ON/15)

6 (a) Combination of staff and task number // ~~the pair of numbers // the pair of random numbers~~ [1]

//there will be duplicates /repeats//some staff tasks will not be generated [1]

(b)

(i) 04 // 03 [1]
(ii) 27 // 28 [1]
(iii) 20 [1]
(iv) 11 / 12 [1]

(c) (i) Zero [1]

(ii) Completed <> 60 // NewStaffTask = FALSE [1]

Allow: Inclusion of the WHILE

(iii) Determines whether this combination of StaffNum and TaskNum has been completed [1]
Assigns value TRUE if not already generated [1]
Flags that this is the first time this staff + task has been selected/to exit the loop [1]
Outputs the new staff + task number [1]
[MAX 3]

(iv) TaskGrid : ARRAY[1:5, 1:12] OF BOOLEAN
1 mark | 1 mark [2]