Algorithm design and problem-solving – 2021 IGCSE 0478

1. Nov/2021/Paper_21/No.1(a),(b),(c),(e)

DO NOT attempt Tasks 1, 2 and 3 now.

Use the pre-release material and your experience from attempting the following tasks before the examination to answer Question 1.

Pre-release material

A holiday park has eight squash courts that can be booked for an hour at a time. The first booking is from 08:00 to 09:00 and the last booking is from 17:00 to 18:00. All bookings start on the hour and bookings can only be made on the same day that the squash court is used. A screen displays today's date and how many squash courts are available for each hour.

When a booking is made, the name of the guest is recorded together with their mobile phone number. Once the squash court is booked, the guest is shown the court number together with a unique 4-digit code that can be used to unlock the squash court. Each booking is for one squash court for one hour. The 4-digit code must be different for each booking.

Write and test a program or programs for a computer system to manage the daily squash court bookings.

- Your program or programs must include appropriate prompts for the entry of data; data must be validated on entry.
- Error messages and other output need to be set out clearly and understandably.
- All variables, constants and other identifiers must have meaningful names.

You will need to complete these **three** tasks. Each task must be fully tested.

Task 1 – at the start of the day

Write a program to set up arrays to record the following for each hour:
- whether a squash court is booked or available
- the name of the guest
- the mobile phone number of the guest
- the unique 4-digit code for the booking.

Set up a screen to display the court availability at the start of the day.

Task 2 – making a squash court booking

Check if there is a squash court available at the time requested. If a squash court is available, record the guest's name and mobile phone number. Mark the squash court as booked for that hour. Generate and record the unique 4-digit code for the booking. Display the mobile phone number for the guest to check, display the court number and the 4-digit code for the guest to remember. Display the updated court availability, showing an hour as fully booked if all the squash courts are now booked for that hour.

Task 3 – at the end of the day

Calculate the total number of squash court bookings. Find the hour(s) and court(s) with the most bookings. Display this information.

All variables, constants and other identifiers must have meaningful names.

(a) Identify **one** constant that you could have used for **Task 1**. Give the value that would be assigned to this constant. State the use of this constant.

Constant .....

Value .....

Use .....

.....

[3]

(b) Describe the arrays that you have set up in **Task 1** to record today's data about the squash courts.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(c) Explain how your program generates a unique 4-digit code for each booking.

.....

.....

.....

.....

.....

..... [3]

(e) Explain how your program completed the part of **Task 3** which calculates the total number of squash court bookings, finds the court(s) with the most bookings and displays this information. Include any programming statements that you have used and fully explain the purpose of each statement.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [4]

2. Nov/2021/Paper_21/No.2(a),(c)

An algorithm has been written in pseudocode to generate 50 positive random integers with values less than or equal to 100. These random integers are stored in the array `RandNum[ ]`

The function `Rand(X, Y)` generates a random integer greater than or equal to `X` and less than `Y`. For example, `Rand(1, 4)` generates `1` or `2` or `3`.

```
1 Count ← 0
2 REPEAT
3     RandNum[Counter] ← Rand(1, 100)
4     Count ← Count + 2
5 UNTIL Count <= 50
```

(a) Find the **four** errors in the pseudocode and write a correction for each error.

Error 1 .....

Correction .....

.....

Error 2 .....

Correction .....

.....

Error 3 .....

Correction .....

.....

Error 4 .....

Correction .....

.....

.....

[4]

(c) Identify another loop structure available in pseudocode.

..... [1]

3. Nov/2021/Paper_21/No.3(a),(b)

A program has been written to check the value of a measurement. The measurement must be a positive number and given to three decimal places, for example, `3.982`

(a) (i) State suitable examples of normal and erroneous test data that could be used to test this program. For each example give the reason for your choice of test data.

Normal test data example .....

Reason .....

.....

Erroneous test data example .....

Reason .....

.....

[4]

(ii) Explain why two pieces of boundary test data are required for this program.
Give an example of each piece of boundary test data.

.....

.....

.....

.....

..... [3]

(b) Explain why verification is needed and how verification could be performed by this program.

.....

.....

.....

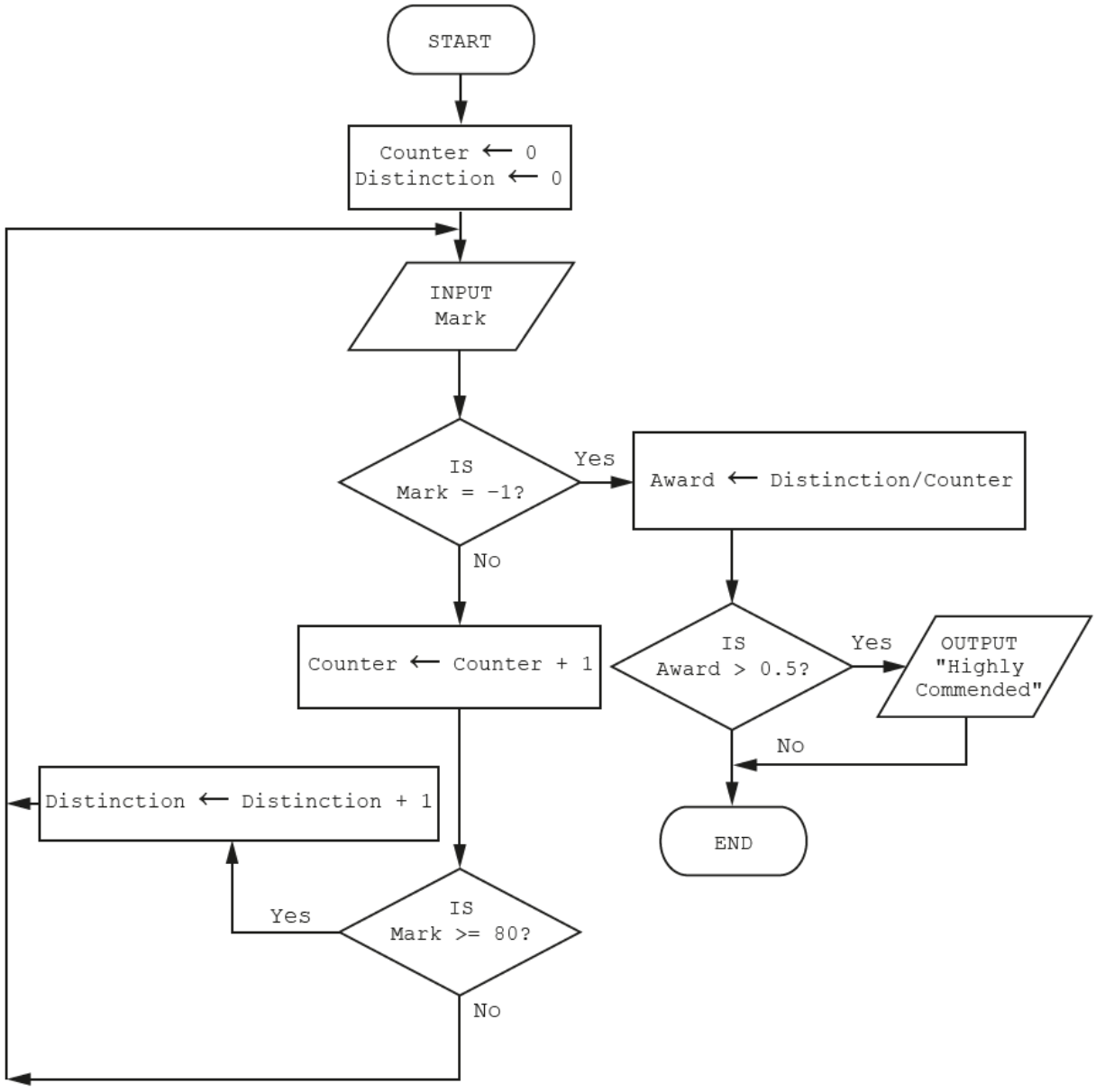
.....

.....

..... [3]

4. Nov/2021/Paper_21/No.4

The algorithm shown by this flowchart allows the input of examination marks for a class of students. A mark of –1 ends the process. If a mark is 80 or over then a distinction grade is awarded. The number of distinctions for the whole class is calculated. If this is over 50% of the class, the class is awarded a highly commended certificate.



Complete a trace table for the algorithm using this input data:
88, 74, 60, 90, 84, 87, 95, 72, 84, 66, –1

Counter	Distinction	Mark	Award	OUTPUT

[5]

5. Nov/2021/Paper_22/No.1(a),(b)

DO NOT attempt Tasks 1, 2 and 3 now.

Use the pre-release material and your experience from attempting the following tasks before the examination to answer Question 1.

Pre-release material

An integrated transport system has been designed to reduce the need for privately owned vehicles. A vehicle is booked to take a passenger from home to a start station, from where they will travel to an end station. A vehicle at the end station will take the passenger to their destination. Each stage of the journey has a price code to represent the distance travelled. The prices for each stage are shown:

Home to start station	
Code	Price ($)
C1	1.50
C2	3.00
C3	4.50
C4	6.00
C5	8.00

Start station to end station	
Code	Price ($)
M1	5.75
M2	12.50
M3	22.25
M4	34.50
M5	45.00

End station to destination	
Code	Price ($)
F1	1.50
F2	3.00
F3	4.50
F4	6.00
F5	8.00

To book a journey, a passenger will enter a code for each stage and the start time of their journey. The total price is calculated by adding together the price for each of the three stages. The total price will be reduced by 40% when the start time of the journey is after 10:00.

Write and test a program or programs for the integrated transport booking system.
- Your program or programs must include appropriate prompts for the entry of data; data must be validated on entry.
- Error messages and other output need to be set out clearly and understandably.
- All variables, constants and other identifiers must have meaningful names.

You will need to complete these **three** tasks. Each task must be fully tested.

Task 1 – setting up the booking system
Write a program to set up arrays to record the following:
- codes and prices for each of the three stages
- passenger accounts that include a unique passenger account number and name
- bookings that include a unique passenger account number, a start time of the journey, a code for each stage of the journey, and a unique booking number for the journey.

Store the data for the code and price for each stage.

Task 2 – using the booking system
Extend **Task 1** to achieve the following:
- Allow passengers to open an account by generating a unique passenger account number and storing it along with their name in the arrays.
- Allow passengers to make a booking by first entering their unique passenger account number, the start time of their journey, and a code for each stage of their journey. Check if the passenger account number already exists.
- Generate a unique booking number for the journey.
- Calculate the total price of the journey, without any discount, and store the journey details.

Task 3 – applying a discount and checking the entry
Extend **Task 2** to check the start time of the journey and if it is after 10:00, apply a 40% discount to the total price.
Display the total price and booking details for the passenger to check, and allow them to either confirm the details are correct or start again.

All variables, constants and other identifiers must have meaningful names.

(a) (i) Identify **one** variable you could have used for **Task 2** and state its use.

Variable .....

Use .....

.....

.....

[2]

(ii) Describe the arrays you could have used in **Task 1**. Include the name, data type, use and sample data for each array.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(b) Describe how you could validate the data entry for the input of the codes for the different stages of the journey in **Task 2**.

.....

.....

.....

.....

.....

..... [3]

6. Nov/2021/Paper_22/No.2

Tick (✓) one box in each row to identify if the statement is about validation, verification or neither.

Statement	Validation (✓)	Verification (✓)	Neither (✓)
a check where data is re-entered to make sure no errors have been introduced during data entry			
an automatic check to make sure the data entered has the correct number of characters			
a check to make sure the data entered is sensible			
a check to make sure the data entered is correct			

[3]

7. Nov/2021/Paper_22/No.3

A program checks that the data entered is between 1 and 100 inclusive.

Identify **one** piece of normal, extreme and erroneous test data for this program, and give a reason for each.

Normal test data .....

Reason .....

.....

.....

Extreme test data .....

Reason .....

.....

.....

Erroneous test data .....

Reason .....

.....

.....

[6]

8. Nov/2021/Paper_22/No.4(a)

The pseudocode algorithm should work as a calculator and output the result.

```
1   Continue ← 1
2   WHILE Continue = 0
3     OUTPUT "Enter 1 for +, 2 for -, 3 for * or 4 for /"
4     INPUT Operator
5     OUTPUT "Enter the first value"
6     INPUT Value1
7     OUTPUT "Enter the second value"
8     OUTPUT Value2
9     IF Operator
10      1: Answer ← Value1 + Value2
11      2: Answer ← Value1 - Value2
12      3: Answer ← Value1 * Value2
13      4: Answer ← Value1 / Value2
14    ENDCASE
15    OUTPUT "The answer is ", Value1
16    OUTPUT "Do you wish to enter more values (Yes or No)?"
17    INPUT MoreValues
18    IF MoreValues = "No"
19      THEN
20        Continue ← 1
21    ENDIF
22  UNTIL Continue = 0
```

(a) Find the **five** errors in the pseudocode and suggest a correction for each error.

Error 1 .....

Correction .....

.....

Error 2 .....

Correction .....

.....

Error 3 .....

Correction .....

.....

Error 4 .....

Correction .....

.....

Error 5 .....

Correction .....

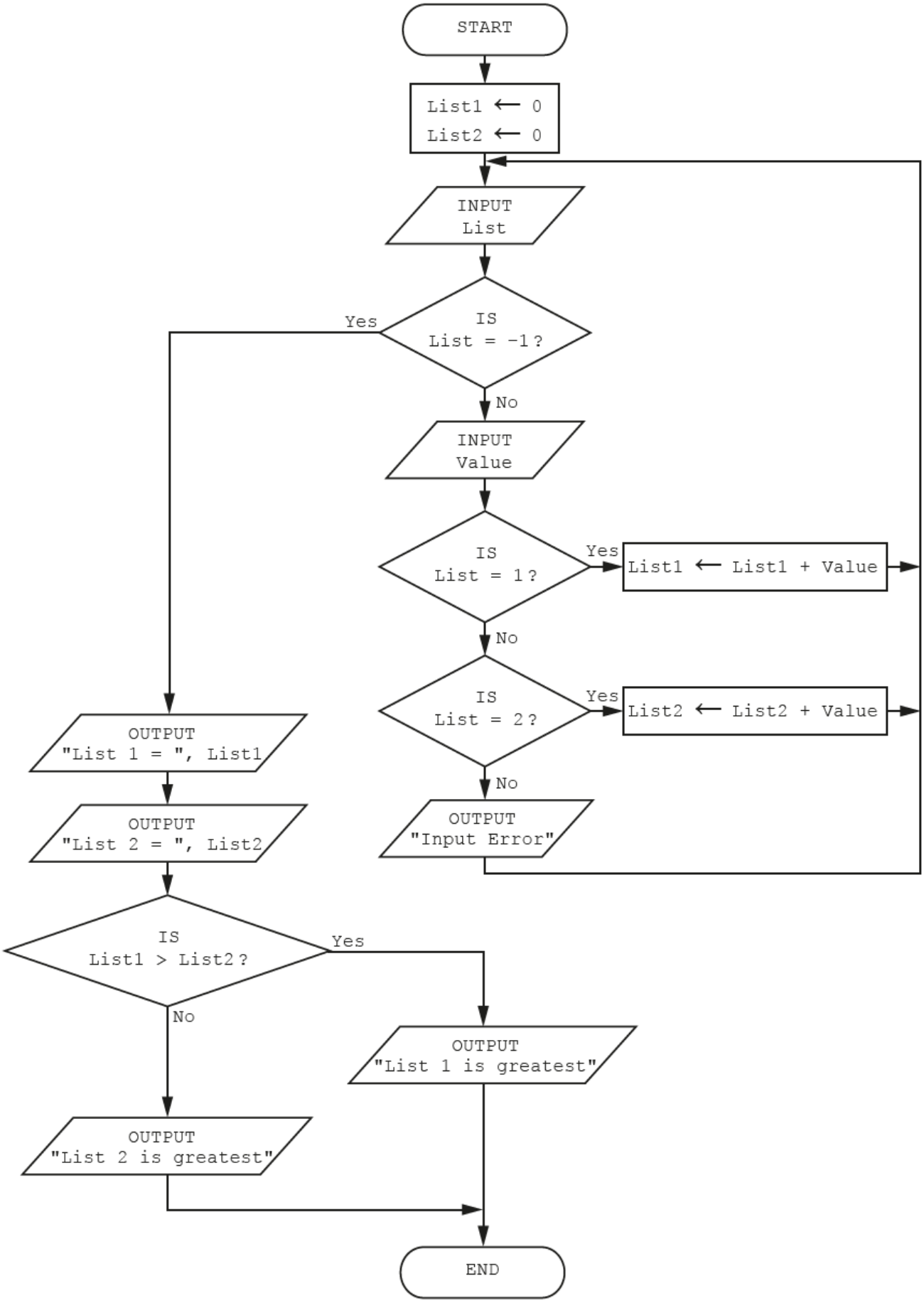
.....

[5]

9. Nov/2021/Paper_22/No.5

The flowchart represents an algorithm.

The algorithm will terminate if –1 is entered at the `List` input.


START
List1 ← 0
List2 ← 0
INPUT List
IS List = -1 ?
Yes
No
INPUT Value
IS List = 1 ?
Yes
List1 ← List1 + Value
No
IS List = 2 ?
Yes
List2 ← List2 + Value
No
OUTPUT "Input Error"
OUTPUT "List 1 = ", List1
OUTPUT "List 2 = ", List2
IS List1 > List2 ?
Yes
No
OUTPUT "List 1 is greatest"
OUTPUT "List 2 is greatest"
END

Complete the trace table for the algorithm using this input data:

2, 77, 2, 16, 1, 35, 2, –7, 5, 18, 1, 11, 1, 12, 2, 20, –1, 18

List	Value	List1	List2	OUTPUT

[5]

10. Nov/2021/Paper_23/No.1(a),(b),(c),(e)

DO NOT attempt Tasks 1, 2 and 3 now.

Use the pre-release material and your experience from attempting the following tasks before the examination to answer Question 1.

Pre-release material

A cruise ship has a speciality restaurant where tables can be booked for any of the three sessions: lunch, early dinner or late dinner. A booking for a table can only be made on the day. There are twenty tables available to book in the restaurant. Today's date and how many tables are available for lunch, early dinner and late dinner are displayed on a screen at the entrance to the restaurant.

Only one table can be booked at a time. When a booking is made, the name of the passenger making the booking is recorded, together with their cabin number and any special dietary requirements.

Write and test a program or programs for a computer system to manage the daily restaurant bookings.

- Your program or programs must include appropriate prompts for the entry of data; data must be validated on entry.
- Error messages and other output need to be set out clearly and understandably.
- All variables, constants and other identifiers must have meaningful names.

You will need to complete these **three** tasks. Each task must be fully tested.

Task 1 – at the start of the day

Write a program to set up the screen display at the start of the day showing the date and how many tables are available for lunch, early dinner and late dinner. Bookings are to be stored in three separate arrays: lunch, early dinner and late dinner. Initialise further arrays to record for each table booked: the name of the passenger making the booking, their cabin number and any special dietary requirements.

Task 2 – making a table booking at the restaurant

Check if there is a table available for the session requested. If a table is available, record the passenger's name, cabin number and any special dietary requirements. Mark the table as booked for that session. Display the name and cabin number for the passenger to check. Update the screen display and mark a session as fully booked if all the tables are now booked.

Task 3 – special dietary requirements

The recording of special dietary requirements is confusing the restaurant staff. It has been decided to use fixed options instead of a description. Only one option can be chosen for each booking.

The options are:

- gluten-free
- vegetarian
- vegan
- diabetic
- none.

Update **Task 2** to allow for this.

Update **Task 2** to count and display how many tables have vegetarian or vegan diners during the day.

All variables, constants and other identifiers must have meaningful names.

(a) Identify **one** constant that you could have used for **Task 1**. Give the value that would be assigned to this constant. State the use of this constant.

Constant .....

Value .....

Use .....

.....

[3]

(b) Describe the arrays that you have set up in **Task 1** to record today's data about the restaurant tables.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(c) Sometimes the restaurant has fewer tables available for a session.

Explain how you would change your **Task 1** program to allow input of the number of tables available for a session.

.....

.....

.....

.....

.....

..... [3]

(e) Explain how your program for **Task 3** updated **Task 2**.

Include any programming statements used and fully explain the purpose of each statement.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

11. Nov/2021/Paper_23/No.2(a)

An algorithm has been written in pseudocode to generate 50 positive random integers with values less than or equal to 100. These numbers are stored in the array `NumRand[]`

The function `RandUp(X,Y)` generates a random integer greater than `X` and less than or equal to `Y`
For example, `RandUp(1,4)` generates `2` or `3` or `4`

```
1 Count ← 0
2 WHILE Counter > 50 DO
3     NumRand[Counter] ← RandUp(1,100)
4     Counter ← Counter - 2
5 ENDWHILE
```

(a) Find the **four** errors in the pseudocode and write a correction for each error.

Error 1 .....

Correction .....

.....

Error 2 .....

Correction .....

.....

Error 3 .....

Correction .....

.....

Error 4 .....

Correction .....

.....

.....

[4]

12. Nov/2021/Paper_23/No.3(a),(b)

A program has been written to check the length and content of a password. The password must be eight or more characters long and contain at least one special character, for example, `Secret!*!`

(a) **(i)** State suitable examples of normal and erroneous test data that could be used to test this program. For each example give the reason for your choice of test data.

Normal test data example .....

Reason .....

.....

Erroneous test data example .....

Reason .....

.....

[4]

(ii) Explain why two pieces of boundary test data are required for this program.
Give an example of each piece of boundary test data.

.....

.....

.....

.....

..... [3]

(b) Describe **two** methods of verification that could be used to verify this data as it is input.

Method 1 .....

.....

.....

Method 2 .....

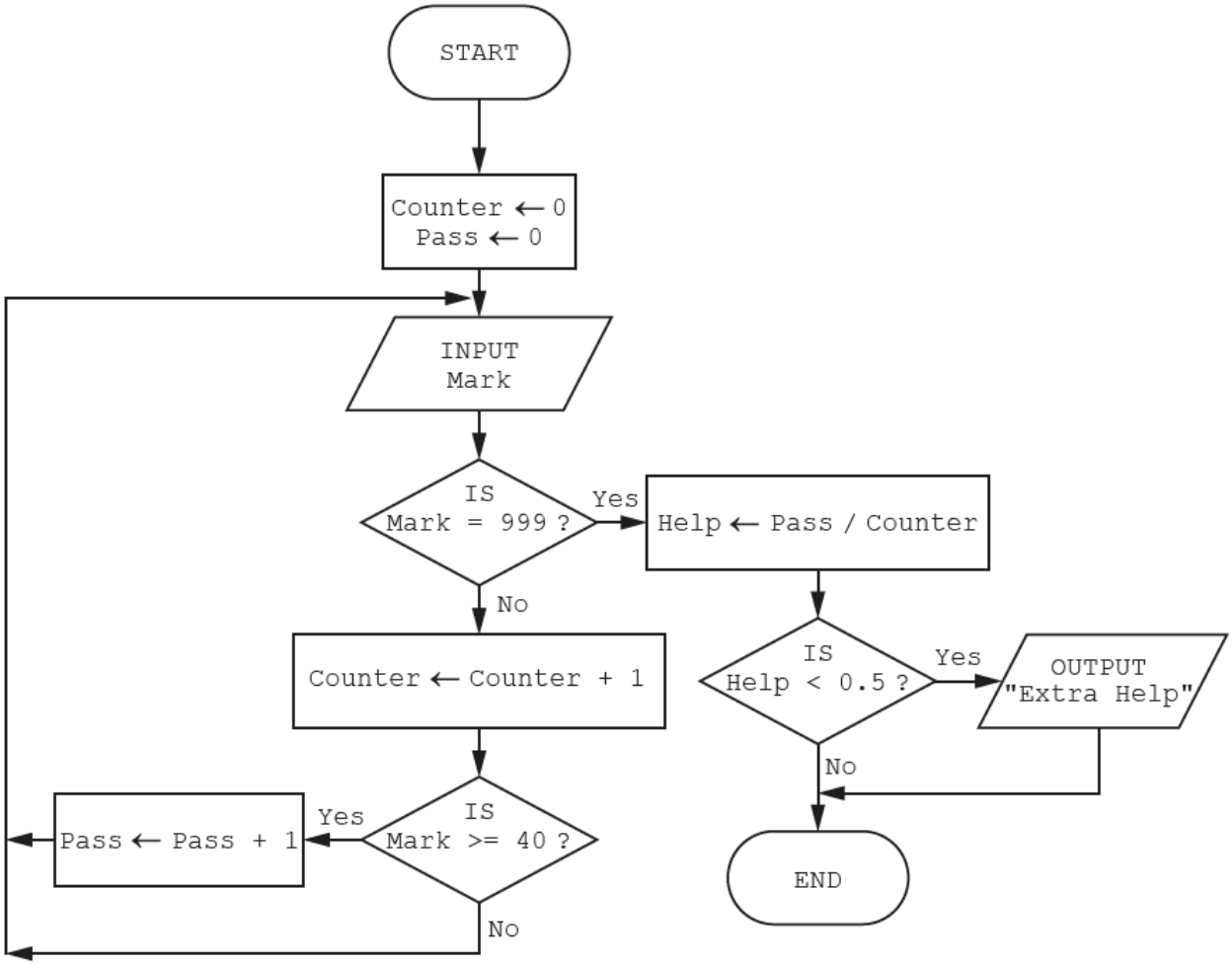
.....

.....

[4]

13. Nov/2021/Paper_23/No.4

The algorithm, shown by this flowchart, allows the input of examination marks for a class of students. A mark of 999 ends the process. If a mark is 40 or over then a pass grade is awarded. The number of pass grades is calculated for the whole class. If this is under 50% of the class, the class is offered extra help.



Complete a trace table for the algorithm using this input data:
88, 24, 60, 30, 44, 17, 25, 22, 54, 6, 999, −1

Counter	Pass	Mark	Help	OUTPUT

[5]

14. March/2021/Paper_22/No.1(a),(b),(d),(e)

DO NOT attempt Tasks 1, 2 and 3 now.

Use the pre-release material and your experience from attempting the tasks before the examination to answer Question 1.

Pre-release material

A program is needed for a quiz to help younger students to practise their multiplication tables. There needs to be two ways of using the quiz; testing and learning.

Testing: the student is given **one** attempt at answering each question and the score is calculated for the whole test.

Learning: the student is given up to **three** attempts to get their answer to each question correct. There is no scoring.

A student can choose which multiplication table, from 2 to 12, to use for the quiz. There are five questions in each quiz, each question must use the chosen multiplication table and a different whole number (from 1 to 12) as the multiplier.

Write and test a program or programs for a multiplication tables quiz.

- Your program or programs must include appropriate prompts for the entry of data; data must be validated on entry.
- Error messages and other output need to be set out clearly and understandably.
- All variables, constants and other identifiers must have meaningful names.

You will need to complete these **three** tasks. Each task must be fully tested.

Task 1 – Testing a student

Students enter their name and choice of multiplication table. Each question is displayed on the screen one at a time, for example:

Question 1 **2 X 7 =**

Students enter their answer and move on to the next question. A running total of correct answers (score) is kept. At the end of the quiz the student's name and score are displayed with a personalised message related to the score, for example:

Aarav your score is 5/5 **Well done full marks**

Diya your score is 3/5 **Have another practice**

Task 2 – Student learning

Students enter their name and choice of multiplication table. Each question is displayed on the screen as in **Task 1**. If an answer is correct, a personalised message containing the student's name confirms this, the quiz then moves to the next question. If an answer is incorrect, a personalised message containing the student's name and a hint is displayed, for example:

Aarav your answer is too large

Up to three attempts are offered to get each answer correct. After the third incorrect attempt, the correct answer is displayed and the quiz moves on to the next question.

Task 3 – Varying the quiz

Modify **Task 1** to allow students to choose how many questions they would like in the test and if they would like a 'mixed' set of questions. A 'mixed' set means that each question can be from a different multiplication table; from 2 to 12.

All variables, constants and other identifiers must have meaningful names.

(a) Identify the variable that you used to store the student's answer in **Task 1**. Give the most appropriate data type for this variable. Explain how your program ensured that any data entered for the answer was valid.

Variable .....

Data type .....

Validation .....

.....

.....

.....

[4]

(b) Identify and give the data type of a **different** variable, that you could have used in **Task 2**. State the use of this variable in **Task 2**.

Variable .....

Data type .....

Use .....

.....

[3]

(d) Explain how your program completed **Task 3**.
Include any programming statements that you have added to **Task 1** and fully explain the purpose of each statement.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(e) Explain how you could alter **Task 1** to change the quiz to:

- display three alternative answers for each question
- allow the student to choose one of these answers

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [3]

15. March/2021/Paper_22/No.2

An algorithm has been written in pseudocode to:

- input 25 positive whole numbers less than 100
- find and output the largest number
- find and output the average of all the numbers

```
01  A ← 0
02  B ← 0
03  C ← 0
04  REPEAT
05    REPEAT
06      INPUT D
07    UNTIL D > 0 AND D < 100 AND D = INT(D)
08    IF D > B
09      THEN
10        B ← D
11    ENDIF
12    C ← C + D
13    A ← A + 1
14  UNTIL A >= 25
15  E ← C / A
16  OUTPUT "Largest number is ", B
17  OUTPUT "Average is ", E
```

(a) Give the line number for the statements showing:

Totalling .....

Counting .....

Range check .....

Calculating the average .....

[4]

(b) State an example for each type of test data needed to test the input of the number:

Normal test data example .....

Erroneous/abnormal test data example .....

Extreme test data example .....

[3]

(c) The algorithm needs to be changed to include finding and outputting the smallest number input. Describe how you would change the algorithm.

.....

.....

.....

.....

.....

.....

.....

.....

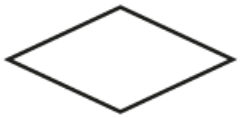
.....

..... [4]

16. March/2021/Paper_22/No.3

Four pseudocode statements and three flowchart symbols are shown.

Draw a line from each pseudocode statement to its correct flowchart symbol.

Pseudocode statement	Flowchart symbol
`IF X > 12`	
`INPUT X`	
`X ← Y + Z`	
`OUTPUT X`	

[4]

17. March/2021/Paper_22/No.4

This algorithm accepts weights of bags of cookies. Any cookie bag weighing between 0.9 and 1.1 kilograms inclusive is acceptable. Underweight bags weigh less than 0.9 kilograms and overweight bags weigh more than 1.1 kilograms. An input of a negative number stops the process. Then the total number of bags, the number of overweight bags and the number of underweight bags weighed are output.

```
Accept ← 0
Over ← 0
Under ← 0
OUTPUT "Enter weight of first cookie bag"
INPUT BagWeight
WHILE BagWeight > 0
    IF BagWeight > 1.1
      THEN
        Error ← 1
      ELSE
        IF BagWeight < 0.9
          THEN
            Error ← 2
          ELSE
            Error ← 0
        ENDIF
    ENDIF
    CASE Error OF
      0 : Accept ← Accept + 1
      1 : Over ← Over + 1
      2 : Under ← Under + 1
    ENDCASE
  OUTPUT "Weight of next bag?"
  INPUT BagWeight
ENDWHILE
Total ← Accept - Over - Under
OUTPUT "Number of bags weighed ", Total
OUTPUT "Number overweight ", Over
OUTPUT "Number underweight ", Under
```

(a) Complete a trace table for the given algorithm using this input data:
1.05, 0.99, 1.2, 0.85, 1.1, 0.9, 1.5, 0.95, 1.05, 1.00, 1.07, 0.89, –10

BagWeight	Accept	Over	Under	Error	Total	**OUTPUT**

[7]

(b) There is an error in this algorithm.
Identify the error and write the corrected pseudocode statement.

Error .....

Correction .....

.....

[2]

18. June/2021/Paper_21/No.1(a),(b),(d)

DO NOT attempt Tasks 1, 2 and 3 now.

Use the pre-release material and your experience from attempting the tasks before the examination to answer Question 1.

Pre-release Material

A system is required to record and count votes for candidates in school council elections. The voting system will allow for one representative to be elected from a tutor group. The school has between 28 and 35 students in each tutor group, five year groups named Year 7 to Year 11, and there are six tutor groups in each year group. Tutor group names are their year group followed by a single letter e.g. 7A, 7B, etc.

All students are allowed to vote in the system. Each student may only vote once for a representative from their tutor group in the election.

Write and test a program or programs for the voting system.

- Your program or programs must include appropriate prompts for the entry of data; data must be validated on entry.
- Error messages and other output need to be set out clearly and understandably.
- All variables, constants and other identifiers must have meaningful names.

You will need to complete these **three** tasks. Each task must be fully tested.

Task 1 – Setting up the voting system to allow a tutor group to elect a representative.
Write a program to:

- allow the tutor to enter the name of the tutor group
- allow the tutor to enter the number of students in the tutor group
- allow the tutor to enter the number of candidates in the election; maximum of four candidates
- allow the tutor to enter the names of the candidates and store them in a suitable data structure
- allow each student to input their vote or to abstain
- count the votes for each candidate and student abstentions.

When all students have voted, display the name of the tutor group, the votes for each candidate and the name of the candidate who has won the election. If there is a tie for first place, display all candidates with the equal highest number of votes.

Task 2 – Checking that students only vote once.
Each student is given a unique voter number by their teacher.
Extend **Task 1** to achieve the following:

- Allow students to enter their unique voter number before casting their vote.
- Check whether the student has already voted:
  - if so, supply a suitable message and do **not** allow them to vote.
  - if not, store the unique voter number, but **not** their vote, in a suitable data structure, and add their vote to the relevant candidate count or abstention.

Task 3 – Showing statistics and dealing with a tie.
Extend **Task 2** to achieve the following:

- Calculate the percentage of the votes that each candidate received from the number of votes cast, excluding abstentions.
- Display the name of each candidate, the number of votes and the percentage of votes they received from the number of votes cast, excluding abstentions.
- Display the total number of votes cast in the election and the number of abstentions.
- In the event of a tie, allow the election to be immediately run again, with only the tied candidates as candidates, and all the students from the tutor group voting again.

(a) All variables, constants and other identifiers must have meaningful names.

(i) Identify **one** constant you could have used for **Task 1**, give the value that would be assigned to it and its use.

Constant .....

Value .....

Use .....

.....

.....

[3]

(ii) Identify **one** variable and **one** array you could have used for **Task 1**.
Explain the use of each one.

Variable .....

Use .....

.....

.....

Array .....

Use .....

.....

.....

[4]

(b) Explain how you should change your program in **Task 1** to allow a tutor to enter up to eight candidates for the election.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(d) Explain how your program completes these parts of **Task 3**:

- Calculate the percentage of the votes that each candidate received from the number of votes cast, excluding abstentions.
- Display the name of each candidate, the number of votes and the percentage of votes they received from the number of votes cast, excluding abstentions.
- Display the total number of votes cast in the election and the number of abstentions.

Any programming statements used in your answer must be fully explained.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

19. June/2021/Paper_21/No.2

Tick (✓) **one** box in each row to identify if the statement is about **validation**, **verification** or **both**.

Statement	Validation (✓)	Verification (✓)	Both (✓)
Entering the data twice to check if both entries are the same.			
Automatically checking that only numeric data has been entered.			
Checking data entered into a computer system before it is stored or processed.			
Visually checking that no errors have been introduced during data entry.			

[3]

20. June/2021/Paper_21/No.3

Name and describe the most appropriate programming data type for each of the examples of data given. Each data type must be different.

Data: `37`

Data type name .....

Data type description .....

.....

.....

Data: `Cambridge2021`

Data type name .....

Data type description .....

.....

.....

Data: `47.86`

Data type name .....

Data type description .....

.....

.....

[6]

21. June/2021/Paper_21/No.4 (a),(c)

The pseudocode algorithm shown has been written by a teacher to enter marks for the students in her class and then to apply some simple processing.

```
Count ← 0
REPEAT
  INPUT Score[Count]
  IF Score[Count] >= 70
    THEN
      Grade[Count] ← "A"
    ELSE
      IF Score[Count] >= 60
        THEN
          Grade[Count] ← "B"
        ELSE
          IF Score[Count] >= 50
            THEN
              Grade[Count] ← "C"
            ELSE
              IF Score[Count] >= 40
                THEN
                  Grade[Count] ← "D"
                ELSE
                  IF Score[Count] >= 30
                    THEN
                      Grade[Count] ← "E"
                    ELSE
                      Grade[Count] ← "F"
                  ENDIF
              ENDIF
          ENDIF
      ENDIF
  ENDIF
  Count ← Count + 1
UNTIL Count = 30
```

(a) Describe what happens in this algorithm.

.....

.....

.....

.....

.....

.....

.....

..... [3]

(c) Describe how you could change the algorithm to allow teachers to use it with any size of class.

.....

.....

.....

.....

.....

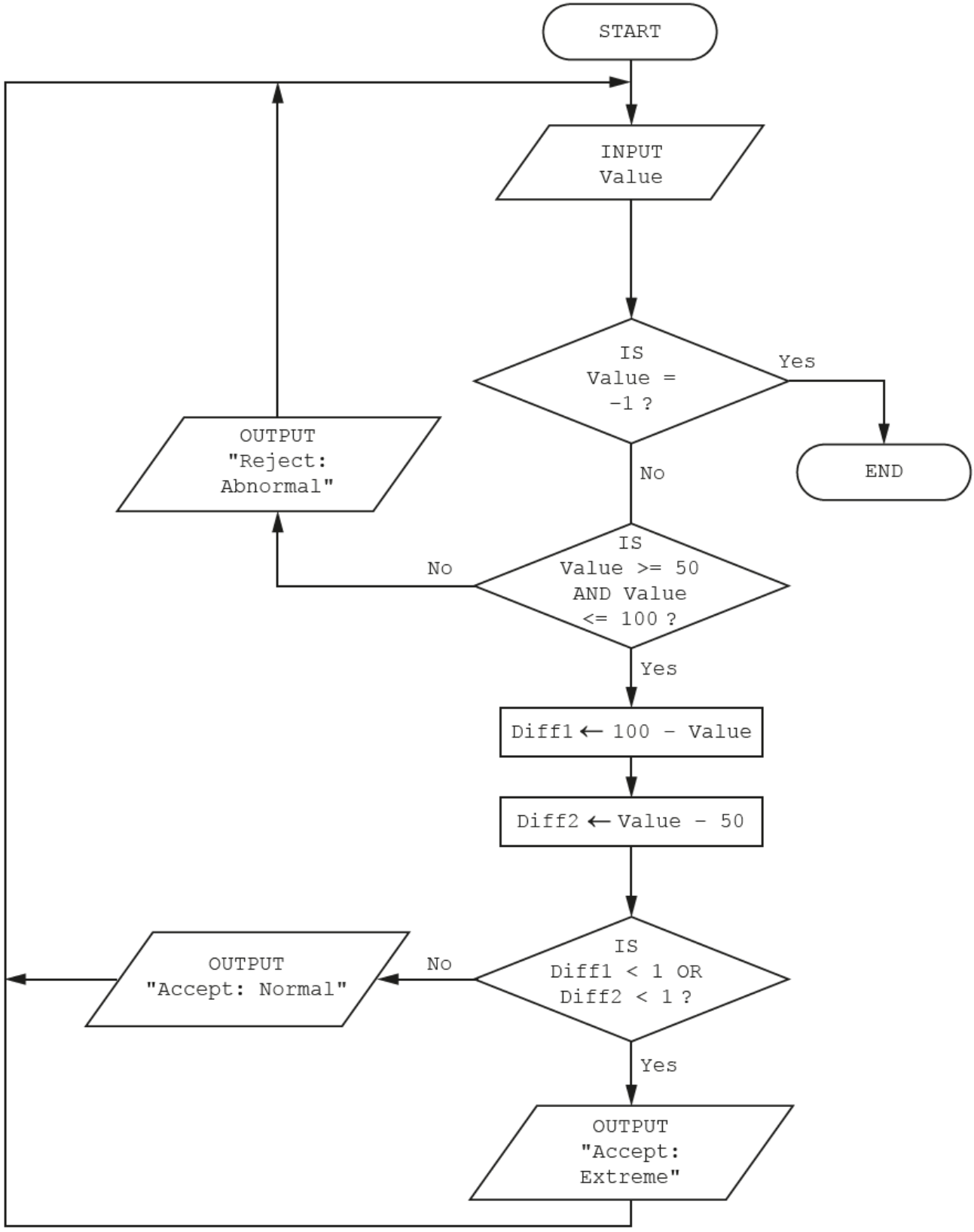
.....

..... [3]

22. June/2021/Paper_21/No.5

The flowchart represents an algorithm.

The algorithm will terminate if –1 is entered.



(a) Complete the trace table for the input data:

50, 75, 99, 28, 82, 150, –1, 672, 80

Value	Diff1	Diff2	OUTPUT

[4]

(b) Describe the purpose of the algorithm.

.....

.....

.....

..... [2]

23. June/2021/Paper_22/No.1(a),(b)(d)

DO NOT attempt Tasks 1, 2 and 3 now.

Use the pre-release material and your experience from attempting the tasks before the examination to answer Question 1.

Pre-release material

An electric mountain railway makes four return trips every day. In each trip the train goes up the mountain and back down. The train leaves from the foot of the mountain at 09:00, 11:00, 13:00 and 15:00. The train returns from the top of the mountain at 10:00, 12:00, 14:00 and 16:00. Each train has six coaches with eighty seats available in each coach. Passengers can only purchase a return ticket; all tickets must be purchased on the day of travel. The cost is $25 for the journey up and $25 for the journey down. Groups of between ten and eighty passengers inclusive get a free ticket for every tenth passenger, provided they all travel together (every tenth passenger travels free). Passengers must book their return train journey, as well as the departure train journey, when they purchase their ticket. Passengers can return on the next train down the mountain or a later train. The last train from the top of the mountain has two extra coaches on it.

The train times are displayed on a large screen, together with the number of tickets still available for each train. Every time a ticket is booked the display is updated. When a train is full, the word 'Closed' is displayed instead of the number of tickets available.

Write and test a program or programs for the electric mountain railway.

- Your program or programs must include appropriate prompts for the entry of data; data must be validated on entry.
- Error messages and other output need to be set out clearly and understandably.
- All variables, constants and other identifiers must have meaningful names.

You will need to complete these **three** tasks. Each task must be fully tested.

Task 1 – Start of the day.
Write a program to set up the screen display for the start of the day. Initialise suitable data structure(s) to total passengers for each train journey and total the money taken for each train journey. Each train journey must be totalled separately. There are four journeys up and four journeys down every day.

Task 2 – Purchasing tickets.
Tickets can be purchased for a single passenger or a group. When making a purchase, check that the number of tickets for the required train journeys up and down the mountain is available. If the tickets are available, calculate the total price including any group discount. Update the screen display and the data for the totals.

Task 3 – End of the day.
Display the number of passengers that travelled on each train journey and the total money taken for each train journey. Calculate and display the total number of passengers and the total amount of money taken for the day. Find and display the train journey with the most passengers that day.

All variables, constants and other identifiers must have meaningful names.

(a) Identify and give the data type and use of **one** array that you could have used for **Task 1**.

Array .....

Data type .....

Use .....

[3]

(b) Describe **two** validation checks that could be used when inputting the number of tickets to buy for **Task 2**. For each validation check give one example of normal data and one example of erroneous data.

Validation check 1 .....

.....

.....

Normal data .....

Erroneous data .....

Validation check 2 .....

.....

.....

Normal data .....

Erroneous data .....

[6]

(d) Explain how your program completed **Task 3**. Include any programming statements that you have used and fully explain the purpose of each statement.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

24. June/2021/Paper_22/No.3

(a) Draw the most appropriate flowchart symbol for each pseudocode statement.

Pseudocode statement	Flowchart symbol
`IF Number = 20`	
`PRINT Number`	
`Number ← Number + 1`	

[3]

(b) State the type of each pseudocode statement. For example, `X ← X + Y` is totalling.

`IF Number = 20` .....

`PRINT Number` .....

`Number ← Number + 1` .....

[3]

25. June/2021/Paper_22/No.4

This algorithm checks passwords.

- Each password must be 8 or more characters in length; the predefined function `Length` returns the number of characters.
- Each password is entered twice, and the two entries must match.
- Either `Accept` or `Reject` is output.
- An input of 999 stops the process.

```
REPEAT
  OUTPUT "Please enter password"
  INPUT Password
  IF Length(Password) >= 8
    THEN
      INPUT PasswordRepeat
      IF Password <> PasswordRepeat
        THEN
          OUTPUT "Reject"
        ELSE
          OUTPUT "Accept"
      ENDIF
    ELSE
      OUTPUT "Reject"
  ENDIF
UNTIL Password = 999
```

(a) Complete the trace table for the algorithm using this input data:
Secret, Secret, VerySecret, VerySecret, Pa55word, Pa55word, 999, 888

`Password`	`PasswordRepeat`	OUTPUT

[3]

(b) Explain how the algorithm could be extended to allow three attempts at inputting the matching password. Any pseudocode statements used in your answer must be fully explained.

.....

.....

.....

.....

.....

.....

.....

..... [4]

26. June/2021/Paper_23/No.1(a),(c),(d)

DO NOT attempt Tasks 1, 2 and 3 now.

Use the pre-release material and your experience from attempting the tasks before the examination to answer Question 1.

Pre-release material

A school would like a system to allow students and staff to show their preference on matters relating to the school, such as a proposal for changing the start and finish times of the school day.

Option	Proposed start and finish times of the school day
A	Start: 08:00 – Finish: 15:00
B	Start: 08:20 – Finish: 15:20
C	Start: 08:40 – Finish: 15:40
D	Start: 09:00 – Finish: 16:00
E	Start: 09:30 – Finish: 16:30

The school has 150 students and 20 members of staff. The system is required to accept preferences, and count and report the results to show student preferences, staff preferences and overall results.

Write and test a program or programs for the system.

- Your program or programs must include appropriate prompts for the entry of data; data must be validated on entry.
- Error messages and other output need to be set out clearly and understandably.
- All variables, constants and other identifiers must have meaningful names.

You will need to complete these **three** tasks. Each task must be fully tested.

Task 1 – Setting up the system and recording preferences.
Set up a system to allow preferences to be recorded for each of five different options, labelled A to E.

- Allow a description for each option to be entered.
- Allow students and staff to enter their unique number before their preferences can be entered (everyone is given a unique number by the school).
- Check if the unique number has already been used:
  - if so, supply a suitable message and do not allow preferences to be entered
  - if not:
    - record that the entered unique number has been used
    - allow preferences from 1 to 5 to be entered for each option (1 is 'strongly agree' and 5 is 'strongly disagree')
    - store the preferences in suitable data structures, keeping student and staff preferences separate.

Task 2 – Totalling the preferences and reporting the results.
Extend **Task 1** to achieve the following:

- Allow the preferences for each of the options to be totalled, keeping student and staff preferences separate.
- Display the results as a list of the options, with the totals given for each one as:
  - student results
  - staff results
  - combined results.

Task 3 – Changing the program to include a counting method.
Extend **Task 2** to achieve the following:

- Count the number of times the preference 1, 'strongly agree', was given for each option, counting student and staff preferences separately.
- Display the results as a list of the options, with the number of times preference 1 was given for each option as:
  - student results
  - staff results
  - combined results.

(a) All variables, constants and other identifiers must have meaningful names.

(i) Identify **one** constant you could have used for **Task 1.** Give the value that would be assigned to the constant and explain its use.

Constant .....

Value .....

Use .....

.....

.....

[3]

(ii) Identify **one** variable you could have used for **Task 1** and explain its use.

Variable .....

Use .....

.....

.....

[2]

(iii) Describe **one** array you could have used for **Task 1**. Include the name, data type, length, sample data and use for that array.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

[4]

(c) Explain how you could change your program in **Task 1** to allow the number of options available to be increased to six.

.....

.....

.....

.....

.....

..... [2]

(d) Explain how your program counts the number of times students enter preference 1 ('strongly agree') for each option (A to E), part of **Task 3**. Any programming statements used in your answer must be fully explained.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [3]

27. June/2021/Paper_23/No.2

Draw a line to connect each **Data Type** to the most appropriate **Description**.

Data Type	Description
Real	Must be a whole number
String	Must be one of two values
Integer	May be any number
Boolean	May contain any combination of characters

[3]

28. June/2021/Paper_23/No.3

Identify a suitable validation check that could be used for each piece of normal test data and describe how it would be used. Each validation check must be different.

Test data for entering an email address: **id27@cambridgeuniversity.com**

Validation check name .....

Description of use .....

.....

.....

Test data for entering a year: **2021**

Validation check name .....

Description of use .....

.....

.....

Test data for entering a name: **Ericson-Bower**

Validation check name .....

Description of use .....

.....

.....

[6]

29. June/2021/Paper_23/No.4

The pseudocode algorithm should allow a user to input the number of scores to be entered and then enter the scores. The scores are totalled, the total is output and the option to enter another set of scores is offered.

```
1   Count ← 0
2   REPEAT
3     FullScore ← 20
4     INPUT Number
5     FOR StoreLoop ← 1 TO Number
6       INPUT Score
7       FullScore ← FullScore
8     UNTIL StoreLoop = Number
9     OUTPUT "The full score is ", FullScore
10    OUTPUT "Another set of scores (Y or N)?"
11    OUTPUT Another
12    IF Another = "N"
13      THEN
14        Count ← 1
15    ENDIF
16  UNTIL Count = 1
```

(a) Identify the **four** errors in the pseudocode and suggest a correction for each error.

Error 1 .....

Correction .....

.....

Error 2 .....

Correction .....

.....

Error 3 .....

Correction .....

.....

Error 4 .....

Correction .....

.....

[4]

(b) Show how you could change the algorithm to store the individual scores in the array `ScoreArray[]`, then find and print the average score once the scores have all been entered.

.....

.....

.....

.....

.....

.....

.....

.....

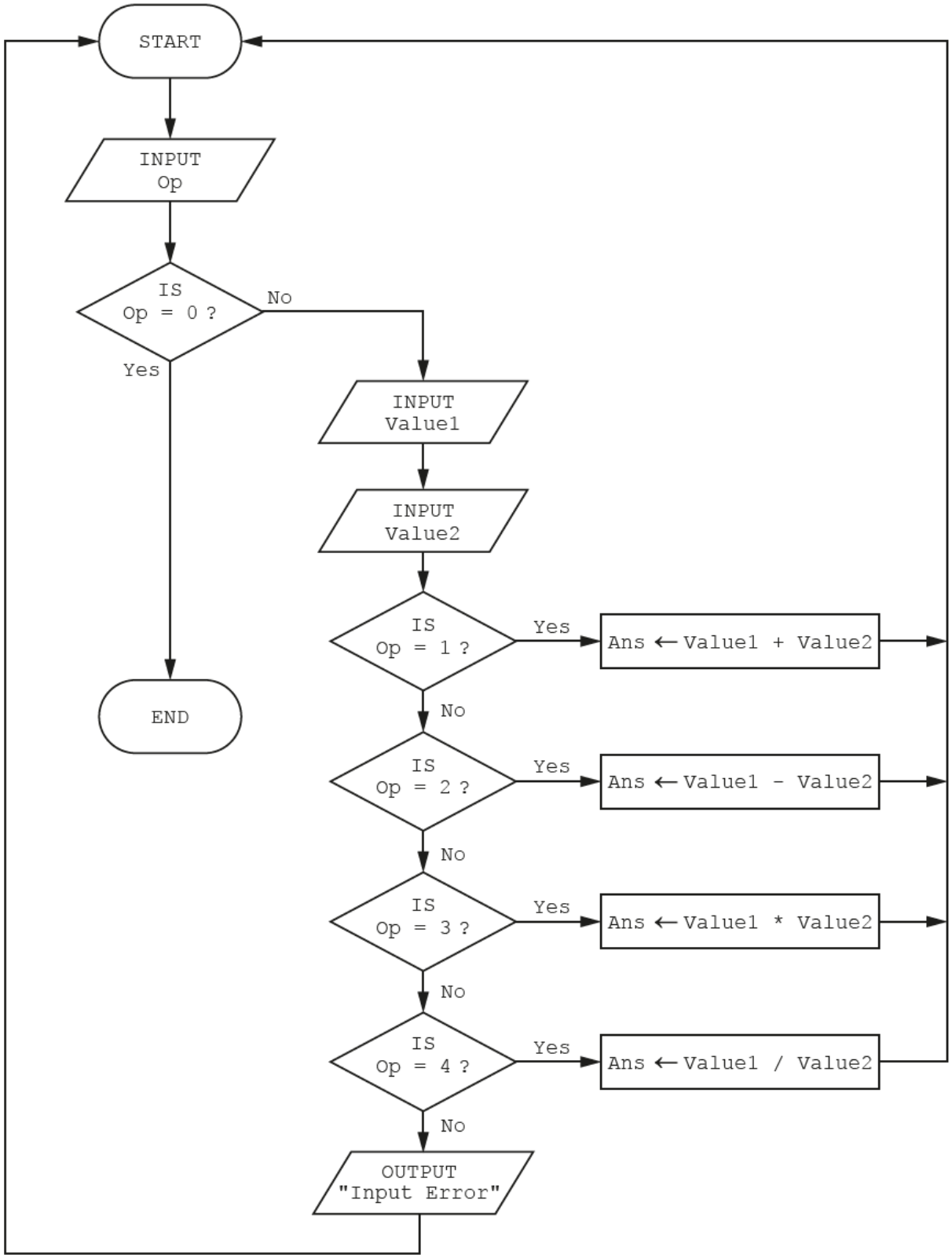
.....

..... [4]

30. June/2021/Paper_23/No.5

The flowchart represents an algorithm.

The algorithm will terminate if 0 is entered at the `Op` input.



(a) Complete the trace table for the algorithm using this input data:

1, 87, 14, 3, 2, 30, 5, 10, 6, 4, 10, 2, 0, 2, 90, 6

Op	Value1	Value2	Ans	OUTPUT

[5]

(b) State the purpose of the algorithm.

.....

.....

.....

..... [1]

(c) Suggest an addition that could be made to the algorithm to make it more useful.

.....

.....

..... [1]