1 **(a)** Algorithms usually consist of three different stages.

One stage is INPUT.

Name the **other** stages.

1 .....

2 .....

[1]

Answer for **a)**

1. Output
2. The process (of generating output)

(b) An algorithm may be documented using different methods. These include structured English, a program flowchart, and pseudocode.

State what a program designer represents using one or more of these methods.

.....

..... [2]

Answer for **b)**

The process which gets converted into machine code, and tells the computer how to handle the input and give out the output to the user.

(c) Programming languages support different data types.

Complete the table by giving four **different** data types together with an example data value for each.

Data type	Example data value

[4]

Answer for **c)**

String - “Khushamdeed Duniya!”

Integer - 198237645

Boolean - False

Double- 3.14

2 A pseudocode algorithm assigns values to three variables as follows:

```
FlagA ← TRUE
FlagB ← FALSE
FlagC ← TRUE
```

Evaluate the expressions given in the following table:

Expression	Evaluates to
`NOT FlagB AND FlagC`	
`NOT (FlagB OR FlagC)`	
`(FlagA AND FlagB) OR FlagC`	
`NOT (FlagA AND FlagB) OR NOT FlagC`	

[2]

Answer for **Q2**

TRUE

FALSE

TRUE

TRU

3 (a) The following pseudocode is an attempt to define an algorithm that takes two numbers as input and outputs the larger of the two numbers.

```
DECLARE A, B : INTEGER
INPUT A
INPUT B
IF A > B
   THEN
      OUTPUT A
   ELSE
      OUTPUT B
ENDIF
```

The algorithm needs to be amended to include the following changes:

1. Input **three** values, ensuring that each value input is unique.
2. Output the average.
3. Output the largest value.

Write the **pseudocode** for the amended algorithm.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

Answer for **3 (a)**

```
DECLARE A, B, C : INTEGER
DECLARE temp: string
INPUT A
INPUT B
INPUT C
WHILE (A = B) OR (B = C) OR (A = C)
        IF A = B THEN
                WHILE (A = B)
                        INPUT A
                END WHILE
        ELSE IF C = B THEN
                WHILE (C = B)
                        INPUT B
                END WHILE
        ELSE IF A = C THEN
                WHILE (A = C)
                        INPUT
                END WHILE
END WHILE
OUTPUT ((A + B + C)/3)
DECLARE sequence[3]
FOR x = 0 to LENGTH(sequence) - 1
        IF sequence [0] > sequence[1]
                temp =   sequence[1]
                sequence[1] = sequence[0]
                sequence[0]  = temp
        ELSE IF sequence [1] > sequence[2]
                temp =   sequence[2]
                sequence[2] = sequence[1]
                sequence[1]  = temp
        ELSE IF sequence [0] > sequence[2]
                temp =   sequence[2]
                sequence[2] = sequence[0]
                sequence[0]  = temp
END FOR
# the following doesn't sort in a perfect sequence, it only makes sure that we get the largest number in the (-1) index

OUTPUT sequence[-1]
```

(b) Complete the pseudocode expressions in the following table.

Use only functions and operators described in the **Appendix**.

Expression	Evaluates to
`"ALARM: " & ..... ("Time: 1202" , .....)`	`"ALARM: 1202"`
`..... ("Stepwise." , ..... , .....)`	`"wise"`
`1.5 * ..... ("OnePointFive")`	`18`
`..... (27.5)`	`"27.5"`
`..... (9, 4)`	`2`

[5]

→ RIGHT , 4

→ MID 4, 7

→ LENGTH

→ NUM_TO_STRING

→ DIV

Appendix

Built-in functions (pseudocode)

Each function returns an error if the function call is not properly formed.

Operators (pseudocode)

Operator	Description
`&`	Concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` produces `"Summer Pudding"`
`AND`	Performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` produces `FALSE`
`OR`	Performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` produces `TRUE`

`LENGTH(ThisString : STRING) RETURNS INTEGER`
returns the integer value representing the length of string `ThisString`

Example: `LENGTH("Happy Days")` returns `10`

`LEFT(ThisString : STRING, x : INTEGER) RETURNS STRING`
returns leftmost `x` characters from `ThisString`

Example: `LEFT("ABCDEFGH", 3)` returns string `"ABC"`

`RIGHT(ThisString: STRING, x : INTEGER) RETURNS STRING`
returns rightmost `x` characters from `ThisString`

Example: `RIGHT("ABCDEFGH", 3)` returns string `"FGH"`

`INT(x : REAL) RETURNS INTEGER`
returns the integer part of `x`

Example: `INT(27.5415)` returns `27`

`MOD(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER`
returns the integer value representing the remainder when `ThisNum` is divided by `ThisDiv`

Example: `MOD(10,3)` returns `1`

`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 string `"BCD"`

`LCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character value representing the lower case equivalent of `ThisChar`
If `ThisChar` is not an upper-case alphabetic character, it is returned unchanged.

Example: `LCASE('W')` returns `'w'`

`DIV(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER`
returns the integer value representing the whole number part of the result when `ThisNum` is divided by `ThisDiv`

Example: `DIV(10,3)` returns `3`

`NUM_TO_STRING(x : REAL) RETURNS STRING`
returns a string representation of a numeric value.
Note: This function will also work if `x` is of type `INTEGER`

Example: `NUM_TO_STRING(87.5)` returns `"87.5"`