Cambridge International AS & A Level

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2025**

INSERT **2 hours**

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

An error will be generated if a function call is not properly formed or if the parameters are of an incorrect type or an incorrect value.

String and character functions

- A string of length 1 may be considered to be either of type CHAR or STRING
- A CHAR may be assigned to, or concatenated with, a STRING
- A STRING of length greater than 1 cannot be assigned to a CHAR

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 string "BCD"
LENGTH(ThisString : STRING) RETURNS INTEGER   returns the integer value representing the length of ThisString   Example: LENGTH("Happy Days") returns 10
TO_UPPER(x : <datatype>) RETURNS <datatype>   <datatype> may be CHAR or STRING   returns an object of type <datatype> formed by converting all characters of x to upper case.   Examples: • TO_UPPER("Error 803") returns "ERROR 803"   • TO_UPPER('a') returns 'A'
TO_LOWER(x : <datatype>) RETURNS <datatype>   <datatype> may be CHAR or STRING   returns an object of type <datatype> formed by converting all characters of x to lower case.   Examples: • TO_LOWER("JIM 803") returns "jim 803"   • TO_LOWER('W') returns 'w'
NUM_TO_STR(x : <datatype1>) RETURNS <datatype2>   returns a string representation of a numeric value.   <datatype1> may be REAL or INTEGER, <datatype2> may be CHAR or STRING   Example: NUM_TO_STR(87.5) returns "87.5"   If x is a negative value, the returned value will be a string beginning with the '−' character.
STR_TO_NUM(x : <datatype1>) RETURNS <datatype2>   returns a numeric representation of a string.   <datatype1> may be CHAR or STRING, <datatype2> may be REAL or INTEGER   Example: STR_TO_NUM("23.45") returns 23.45   If the string begins with the '−' character, the returned value will be negative.

```
IS_NUM(ThisString : <datatype>) RETURNS BOOLEAN
```

returns `TRUE` if `ThisString` represents a valid numeric value.
`<datatype>` may be `CHAR` or `STRING`
Example: `IS_NUM("-12.36")` returns `TRUE`

```
ASC(ThisChar : CHAR) RETURNS INTEGER
```

returns an integer value (the ASCII value) of character `ThisChar`
Example: `ASC('A')` returns 65, `ASC('B')` returns 66

```
CHR(x : INTEGER) RETURNS CHAR
```

returns the character whose integer value (the ASCII value) is `x`
Example: `CHR(65)` returns `'A'`, `CHR(66)` returns `'B'`

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 35.430729

Date functions

Date format is assumed to be DD/MM/YYYY unless otherwise stated.

```
DAY(ThisDate : DATE) RETURNS INTEGER
```

returns the day number from `ThisDate`
Example: `DAY(04/10/2003)` returns 4

```
MONTH(ThisDate : DATE) RETURNS INTEGER
```

returns the month number from `ThisDate`
Example: `MONTH(04/10/2003)` returns 10

```
YEAR(ThisDate : DATE) RETURNS INTEGER
```

returns the year number from `ThisDate`
Example: `YEAR(04/10/2003)` returns 2003

```
DAYINDEX(ThisDate : DATE) RETURNS INTEGER
```

returns the day index number from `ThisDate` where Sunday = 1, Monday = 2 etc.
Example: `DAYINDEX(09/05/2023)` returns 3

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

returns a value of type `DATE` with the value of `<Day>/<Month>/<Year>`
Example: `SETDATE(26, 10, 2003)` returns a date corresponding to 26/10/2003

```
TODAY() RETURNS DATE
```

returns a value of type `DATE` corresponding to the current date.

Text file functions

`EOF(FileName : STRING) RETURNS BOOLEAN`
returns `TRUE` if there are no more lines to be read from file `FileName` will generate an error if the file is not already open in `READ` mode.

Operators

An error will be generated if an operator is used with a value or values of an incorrect type.

`&`	concatenates (joins) two strings. Example: `"Summer" & " " & "Pudding"` evaluates to `"Summer Pudding"` may also be used to concatenate a `CHAR` with a `STRING`
`AND`	performs a logical `AND` on two Boolean values. Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	performs a logical `OR` on two Boolean values. Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	performs a logical `NOT` on a Boolean value. Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	finds the remainder when one number is divided by another. Example: `10 MOD 3` evaluates to `1`
`DIV`	finds the quotient when one number is divided by another. Example `10 DIV 3` evaluates to `3`

Comparison operators

`=`
 	
`>`
 	
`<`
 	
`>=`
 	
`<=`
 	
`<>`
	used to compare two items of the same type.
	evaluates to `TRUE` if the condition is true, otherwise evaluates to `FALSE`
	
	Notes:
	   • may be used to compare types `REAL` and `INTEGER`
	   • may be used to compare types `CHAR` and `STRING`
	   • case sensitive when used to compare types `CHAR` and/or `STRING`
	   • cannot be used to compare two records
	
	Examples:
	   • `"Program" = "program"` evaluates to `FALSE`
	   • `Count = 4` evaluates to `TRUE` when `Count` contains the value 4