Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2015**

PRE-RELEASE MATERIAL

No Additional Materials are required.

This material should be given to candidates on receipt by the Centre.

READ THESE INSTRUCTIONS FIRST

Candidates should use this material in preparation for the examination. Candidates should attempt the practical programming tasks using their chosen high-level, procedural programming language.

DC (LK) 95390/1

This material is intended to be read by teachers and candidates prior to the June 2015 examination for 9608 Paper 2.

Reminders
The syllabus states:

- there will be questions on the examination paper which do not relate to this pre-release material
- you must choose a high-level programming language from this list:
  - Visual Basic (Console Mode)
  - Python
  - Pascal / Delphi (Console Mode)

Questions on the examination paper may ask the candidate to write:

- structured English
- pseudocode
- program code

A program flowchart should be considered as an alternative to pseudocode for the documenting of an algorithm design.

Candidates should be confident with:

- the presentation of an algorithm using either a program flowchart or pseudocode
- the production of a program flowchart from given pseudocode (or the reverse)

Declaration of variables

The syllabus document shows the syntax expected for a declaration statement in pseudocode.

```
DECLARE <identifier> : <data type>
```

It is appreciated that candidates who use Python as their chosen language will not be familiar with the concept of declaring all variables with their data type before they are used.

However, answers using Python will be required, instead of a declaration statement, to include a comment line documenting the identifier name with its intended data type.

The question rubric will clarify this with a wording and answer layout such as:

(i) Write **program code** for the new design.

Visual Basic and Pascal: You should include the declaration statements for variables.
Python: You should show a comment statement for each variable used with its data type.

Programming language .....

.....

.....

Structured English – Variables

An algorithm written in pseudocode requires that all variables have been identified. This may not be the case if the initial attempt at the algorithm design is in structured English. The candidate will then be required to identify the variables from the question rubric.

TASK 1

- Riders complete one stage of a cycle race and their finishing position and time is recorded.
- The time is entered by the user as two integers:
  - number of minutes
  - followed by the number of seconds
- The rider's stage race position is entered.
- The time is converted to seconds.
- If they finish in first, second or third position they get a time bonus, see table below.
- The time bonus is deducted from their race time.

Key focus: Structured English

Position	Time bonus (seconds)
1	20
2	10
3	5

Line numbers may be shown for structured English (and pseudocode) to allow reference to particular line(s)

```
01  INPUT rider name
02  INPUT rider's race time in minutes
03  INPUT rider's race time in seconds
04  INPUT rider's position
05
06  CALCULATE race time in seconds
07  STORE race time in seconds
08  CALCULATE time bonus
09  STORE time bonus
10  SUBTRACT time bonus from race time in
    seconds
11  STORE adjusted race time
12
13  OUTPUT rider's name, position, adjusted
    race time
```

TASK 1.1

A program is to be written to process the data for one rider.

Study the structured English and complete the identifier table below.

Identifier	Data type	Description
`RiderName`	`STRING`	Name of the rider

TASK 1.2

Line `08` in the structured English does not give sufficient detail to write program code from this statement.

Use stepwise refinement to give the detail for Line `08`.

Key focus: Stepwise refinement

Suggested extension tasks

- How could arrays be used to store data from several riders which were input in sequence?
- Draw up test data for eight riders (which include the riders finishing first, second and third).
- Carry out a bubble sort to produce a list of the rider names in finishing order.
- Save the data for the eight riders to a text file `RACE-DATA`.

Operators

Arithmetic operators

In addition to the normal +, –, * and /, candidates should be familiar with the `DIV` and `MOD` operators.

For example: `17 DIV 7` evaluates to `2`

`17 MOD 7` evaluates to `3`

& Operator

The & operator will be used to concatenate two strings.
For example: `"Birthday " & "Cake"`
Evaluates to: `"Birthday Cake"`

Built-in Functions

Any high-level programming language will have many built-in functions for the programmer to use.

It is appreciated that the three programming languages often implement these functions with very different syntax. Candidates should be familiar with the syntax used in their chosen programming language.

If a built-in function is to be used in pseudocode on the examination paper, the function will be shown and explained. Examples of this follow.

Key focus: Built-in function definitions

String handling functions (Pseudocode)

`ONECHAR(ThisString : STRING, Position: INTEGER) RETURNS CHAR`
returns the single character at index position `Position` (counting from the start of the string with value 1) from the string `ThisString`.

For example: `ONECHAR("Hockey", 4)` returns `'k'`

`CHARACTERCOUNT(ThisString : STRING) RETURNS INTEGER`
returns the `number` of characters in string `ThisString.`

For example: `CHARACTERCOUNT("Real Madrid")` returns 11

`SUBSTR(ThisString : STRING, Value1 : INTEGER, Value2 : INTEGER) RETURNS STRING`
returns a sub-string from within `ThisString`.
`Value1` is the start index position (counting from the left, starting with 1).
`Value2` is the final index position.

For example: `SUBSTR("art nouveau", 5, 11)` returns `"nouveau"`

Use of square brackets in a function definition denotes the parameter is optional

```
CONCAT(String1 : STRING, String2 : STRING [, String3 : STRING]) RETURNS
STRING
```

For example: `CONCAT("Los", "Angeles")` returns `"LosAngeles"`
`CONCAT("Los", " ", "Angeles") returns "Los Angeles"`

Conversion between data types (Pseudocode)

```
TONUM(ThisDigit : CHAR) RETURNS INTEGER
```

For example: `TONUM('8')` returns integer `8`

```
TOSTRING(ThisNumber : INTEGER or REAL) RETURNS STRING
```

For example: `TOSTRING(83)` returns `"83"`
`TOSTRING(704.25)` returns `"704.25"`

Using ASCII character codes (Pseudocode)

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

For example: `CHR(65)` returns character `'A'`

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

For example: `ASC('A')` returns integer `65`

TASK 2

Use the functions given to evaluate the following expressions:

1 `CONCAT("Big", " ", "Ben")` .....

2 `CONCAT(CONCAT("Tower", "of"), "London")` .....

3
```
ONECHAR("Camel", 1) & ONECHAR("aardvark", 2)
& ONECHAR("goat", 4)
```
.....

4 `CHARACTERCOUNT("Taj Mahal") + 3` .....

5 `TONUM('8') + TONUM('9')` .....

6
```
x ← "BAMBI"

y ← CHARACTERCOUNT( SUBSTR(x, 3, 5) )
```

State the value assigned to variable y. .....

7
```
MyString ← "The quick brown fox"
```

Using the `SUBSTR` function, write expressions to return:

(a) `"brown"` .....

(b) `"fox"` .....

(c) `"The fox"` .....

State the value produced by:

(d) `SUBSTR(MyString, 5, 9) & SUBSTR(MyString, 1, 3)`

.....

9 Use an ASCII code table to find the value for:

(a) `ASC('B') + ASC('8')` .....

(b) `ASC('Z') – ASC('H')` .....

10 Give the string produced by the following expressions.

(a)
```
CHR(81) & CHR(85) & CHR(69) & CHR(69)
& CHR(78)
```
.....

(b)
```
SUBSTR(CHR(66) & CHR(69) & CHR(65)
& CHR(84), 2, 4)
```
.....

Suggested extension task:

Write more expressions which use the built-in functions given.

BLANK PAGE