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 **October/November 2016**

PRE-RELEASE MATERIAL

5849636695

No Additional Materials are required.

This material should be given to the relevant teachers and candidates as soon as it has been received at 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.

This material is intended to be read by teachers and candidates prior to the November 2016 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)

Note: A mark of zero will be awarded if a programming language other than those listed is used.

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)

Candidates will also benefit from using pre-release materials from previous examinations. These are available on the teacher support site.

There is an **Appendix** starting on page 7 of this document. Some tasks refer you to this information. There will also be a similar appendix at the end of the question paper.

Declaration of variables

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

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

If Python is the chosen language, each variable's identifier (name) and its intended data type must be documented using a comment statement.

Structured English – Variables

An algorithm in pseudocode uses variables, which should be declared. An algorithm in structured English does not always use variables. In this case, the candidate needs to use the information given in the question to complete an identifier table. The table needs to contain an identifier, data type and description for each variable.

Key Focus:

Built-in functions

TASK 1

Work out the value assigned to each of the variables shown below.

You will need to refer to the list of built-in functions in the **Appendix**.

```
x ← "South Africa"

y ← 13 + CHARACTERCOUNT(x)                    y .....

MyString ← "196"

z ← (TONUM(MyString) + 4) / 100               z .....

a  ← "player"

b  ← "happening"

c  ← SUBSTR(a, 1, 4) & SUBSTR(b, 4, 6)        c .....


p  ← INT(RND() + 0.5)
     when the RND function returns 0.50987    p .....
```

TASK 2

Data is received as a stream of characters:

- each stream of characters is a sequence of surnames
- the start and end of a sequence is indicated by two * characters (**)
- the surnames are separated by a # character

A typical character sequence is:

```
**ali#thomas#ghadi#akmar#smith#barija**
```

Write **program code** to:

- input a character sequence (made up of one or more surnames)
- check that the character sequence has a valid format
- calculate and output the number of surnames

TASK 3

A garage records data for each car sale. It also stores the number of repair visits made within a two-year guarantee period.

Data for all cars are to be stored in a text file with the following data items:

- car registration – one alphabetic character, followed by four digit characters
- date sold – format “DD/MM/YYYY”
- number of repair visits

TASK 3.1

Program 1

Create the text file `CARSALES1.TXT` using the data given below.

Each set of car data will be input by the user and then saved to file `CARSALES1.TXT`.

The program will:

- input the car registration, date sold, and number of repairs to date, for the five car sales
- save each data set in the text file

Key Focus:

Text files

```
B5612
12/11/2015
3
G5988
21/12/2015
1
B6009
22/12/2015
1
B6443
08/01/2016
0
G6900
08/01/2016
0
```

Program 2

Confirm that the file has been successfully created.

Write **program code** to output the contents of the file.

Program 3

Search `CARSALES1.TXT` for a particular registration and output the data for that car.

The program will:

- input a car registration
- output:
  - the car data
  - or the message "REGISTRATION NOT FOUND"

Key Focus:

Searching a text file

Program 4

Produce a report that shows cars that have made the specified number, or more, of repair visits.

Write a program to:

- input the number of repair visits
- output:
  - the car registration for all cars with this number of repair visits or higher
  - or the message "NO CARS FOUND"

Program 5

Each time a new car is sold, its data will be added to the file.

Write a program to:

- input the car registration and sales date
- append the car data to `CARSALES1.TXT`

Key Focus:

Appending data to a text file

Program 6

Combine Programs 1 – 5 into a single program with the following menu choices:

1. Create car sales file
2. Search by registration
3. Repairs report
4. Add new car sale
5. Exit

TASK 3.2

Teacher preparation work

Create a text file `CARSALES2.TXT` with around 30 cars. The format of the file is shown below:

The range of dates should be such that the file contains some cars that were sold over two years ago.

```
B5612 12/11/2013 3
G5617 21/12/2013 1
B6003 22/12/2013 0
B6078 08/01/2015 4
G5689 08/01/2015 3
```

```
B7890 03/05/2016 0
G6601 08/05/2016 0
```

Note: The data items for a car are separated using the `<Space>` character.

Make this file `CARSALES2.TXT` available to candidates.

Program 7

Each time a new car is sold, its data will be added to the file.

The program will:

- input the car registration and sales date
- append the car data to `CARSALES2.TXT`

Key Focus:

Appending data to a text file

Using string handling functions

Program 8

Search `CARSALES2.TXT` for sales on a particular date.

The program will:

- input the month number and year
- output the car data for all cars sold during this month

Program 9

A car has just been repaired.

Key Focus:

Amending data in a text file

The program will:

- input the car registration
- search for the car in `CARSALES2.TXT`
- if found:
  - the number of repairs for this car is incremented
  - the amended data for this car is written to the file
- if not found:
  - output a message “CAR NOT FOUND”

Key Focus:

Removing data from a text file

Program 10

Cars over two years old are no longer under guarantee. They need to be removed from the file.

Write a program to remove cars that are no longer under guarantee.

Appendix

Built-in functions (pseudocode)

```
ONECHAR(ThisString : STRING, Position : INTEGER) RETURNS CHAR
```

returns the single character at position `Position` (counting from the start of the string with value 1) from the string `ThisString`.

For example: `ONECHAR("New York", 5)` returns `'Y'`

```
CHARACTERCOUNT(ThisString : STRING) RETURNS INTEGER
```

returns the number of characters in `ThisString`.

For example: `CHARACTERCOUNT("New York")` returns `8`

```
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"`

```
TONUM(ThisString : STRING) RETURNS INTEGER or REAL
```

returns the integer or real equivalent of the string `ThisString`.

For example: `TONUM("502")` returns the integer `502`
`TONUM("56.36")` returns the real number `56.36`

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

returns an integer which is the ASCII character code for the character `ThisCharacter`.

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

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

returns the character represented by ASCII code number `Value`.

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

```
RND() RETURNS REAL
```

returns a random number in the range 0 to 0.99999

For example: `RND()` returns `0.67351`

```
INT(ThisNumber : REAL) RETURNS INTEGER
```

returns the integer part of `ThisNumber`.

For example: `INT(12.79)` returns `12`

Errors

For any function, if the program calls the function incorrectly, the function returns an error.

Concatenation operator

`&` operator – Concatenates two expressions of `STRING` or `CHAR` data type.

For example: `"South" & " " & "Pole"` produces `"South Pole"`
`'B' & "000654"` produces `"B000654"`