General Certificate of Education
Advanced Subsidiary Examination
June 2009

COMPUTING COMP1/PM/PY

Unit 1 Problem Solving, Programming, Data Representation and Practical Exercise

Preliminary Material

To be given to candidates on or after Wednesday 1 April 2009

Information

- This Preliminary Material comprises
  - Instructions to Candidates,
  - a Data File and a
  - Skeleton Program for **Python** which your teacher will supply.

  You must **only** use the version of the Skeleton Program supplied by your teacher.
- Candidates are advised to familiarise themselves with the Preliminary Material before the examination.
- This Preliminary Material will be made available to you again in the examination. You must **not** take any copy of the Preliminary Material or any other material into the examination room. Your teacher will provide you with access to these electronically at the start of the examination.

Instructions to Candidates

Format of the COMP1 Examination

The question paper is divided into four sections and a recommendation is given to candidates as to how long to spend on each section. Below are the recommended timings for the 2009 examination.

SECTION A
You are advised to spend no more than **25 minutes** on this section.
Questions will examine the specification content **not** specific to the **Preliminary Material.**

SECTION B
You are advised to spend no more than **20 minutes** on this section.
Candidates will be required to write a new program from scratch.
Questions will refer to the **Preliminary Material** (excluding the **Skeleton Program**).

SECTION C
You are advised to spend no more than **20 minutes** on this section.
Questions will refer to the **Preliminary Material** (including the **Skeleton Program**) but no coding will be required.

SECTION D
You are advised to spend no more than **55 minutes** on this section.
Questions will use the **Skeleton Program** and may require the **Data File**.

Electronic Answer Document

Answers for all questions for all four sections must be written into the word processed document made available to the candidate at the start of the examination and referred to in the question paper rubric as the **Electronic Answer Document**.

Preparation for the Examination

For your programming language you should ensure that you are familiar with:

- this **Preliminary Material,** including the **Skeleton Program**.

For your programming language, you should be familiar with:

- the built-in functions available for **manipulating string data**
- a method/function for **generating a random number**
- **c**alculating from this random number an integer within a specified range
- **file handling** commands for text files made up of several lines of text.

"**Guess the Word/Phrase Game**"

The **Skeleton Program** in this Preliminary Material is for a game based on the game of "Hangman" in which the *user* is given a certain number of letter guesses to guess a chosen word or phrase.

This exercise will not require the display of a gallows!

The game starts with the input by the *setter* of a word or phrase of **at least 10 characters** chosen from a set of characters consisting of the uppercase letters of the alphabet and the <Space> character. You may assume that the *setter* never sets a word or phrase with more than 20 characters.

The output device, i.e. the screen, then displays a row of asterisks ('*') corresponding to every letter in the word or phrase and a space for every space.

The game will not use words/phrases containing other characters, e.g. hyphens, apostrophes or digits (0, 1, 2, …. 9).

A second person, the *user,* must then enter a letter that they think could be present in the *setter*'*s* word or phrase or if they think that they recognise the word or phrase, they enter this word or phrase.

In each of these cases, what the *user* has done is **make a guess**.

If the letter guess is correct, the row of asterisks displays again with the guessed letter replacing one or more asterisks in the corresponding positions that this letter appears in the word/phrase.

If the word/phrase guess is correct, the game is over.
A message displays which states, "You have guessed correctly."

If either guess is incorrect, the row of asterisks displays again with no change.

Restrictions

Two people are required to play this game, a *setter* and a *user*.

The *setter* inputs the word/phrase to be guessed. The *user*'*s* role is to guess the word/phrase either directly by submitting the word/phrase or indirectly by guessing its letters. The *user* must not have sight of the word/phrase before playing the game.

The **Skeleton Program** does not store or display all letters that the *user* enters (history of letters entered), only those that are correct guesses.

The **Skeleton Program** in its present form does not allow the user to attempt a guess of the complete word/phrase.

The game allows the *user* to make an unlimited number of letter guesses.

The letter case of the *user*'*s* guess must match the word/phrase's letter case which is upper case.

Outline Design

The game begins when a new word or phrase is set. **For the purpose of the design a phrase is considered to consist of one or more words**.

The Structured English description of an algorithm for playing the game is as follows:

```
INPUT word/phrase
check number of characters in word/phrase
IF acceptable
  THEN
    DO
      OUTPUT word/phrase with letters not yet guessed masked by asterisks
      INPUT next letter guess
      IF letter is present
        THEN update values of variables
      ENDIF
    LOOP UNTIL all letters guessed
  ELSE OUTPUT "Not enough letters"
ENDIF
```

Variables

The main variables used are as follows:

Identifier	Data Type	Purpose
`NewPhrase`	`String`	The word/phrase entered by the *setter*
`GuessStatusArray`	`Array[1..20] of Char`	Stores: • an <Asterisk> in each position not yet guessed • the letters in positions that have been guessed
`IndividualLettersArray`[1]	`Array[1..20] of Char`	Stores: • the individual characters from `NewPhrase` • any <Space> character(s) in `NewPhrase` are stored as a <Space>

Note: The programming language used to code the game will determine the letter case for each identifier, and so may not match exactly the identifiers shown in the table above.

[1] The `IndividualLettersArray` is only needed when the programming language does not support direct access to the individual letters of `NewPhrase`.

Guessing a Letter

The data types used by the various programming language **Skeleton Programs** differ slightly.

- All languages store the *setter's* word/phrase in a variable `NewPhrase`

- Languages such as Pascal, C, PHP and C# permit access to individual characters of the string as follows:

  E.g.
  `NewPhrase := 'DERBY COUNTY' ;`
  `NewPhrase[1]` gives `'D'`, `NewPhrase[4]` gives `'B'`.

- Other languages such as Visual Basic.Net are unable to access the individual letters of `NewPhrase` directly.
  An array `IndividualLettersArray` is used to store the letters to permit access to individual letters as follows:

  `NewPhrase = "DERBY COUNTY"`
  Access to the first and fourth letters is made possible as follows:
  `IndividualLettersArray[1]` gives `'D'`.
  `IndividualLettersArray[4]` gives `'B'`.

Figure 1 shows the contents of the `NewPhrase / IndividualLettersArray` for the phrase "COMPUTING EXAM".
`NewPhrase` is used for the languages which support access to individual characters.

All solutions store the letter guesses in array `GuessStatusArray`.
Figure 2 shows the contents of the array `GuessStatusArray` before the user has made any guesses.

Figure 3 shows the contents of the array `GuessStatusArray` after the user has guessed the letters `'E'`, `'A'`, `'W'`, `'P'`, `'C'`, `'U'` in that order.
Note: the unsuccessful guess `'W'` is not stored.

Figure 1

	NewPhrase / IndividualLettersArray
1	C
2	O
3	M
4	P
5	U
6	T
7	I
8	N
9	G
10	<Space>
11	E
12	X
13	A
14	M
:	:
:	:
:	:
20	

Figure 2

	GuessStatusArray
1	*
2	*
3	*
4	*
5	*
6	*
7	*
8	*
9	*
10	<Space>
11	*
12	*
13	*
14	*
:	:
:	:
:	:
20	

Figure 3

	GuessStatusArray
1	C
2	*
3	*
4	P
5	U
6	*
7	*
8	*
9	*
10	<Space>
11	E
12	*
13	A
14	*
:	:
:	:
:	:
20	

Possible Additional Requirement

The **Skeleton Program** does not display every letter that has been entered; it only displays correctly guessed letters.

Consider how the letters entered during the game could be stored. You need only consider the two suggestions shown below.

Suggestion 1
Use a one-dimensional array `LettersGuessedArray`[2], **Figure 4**, with each array cell corresponding to a letter of the alphabet.

Figure 4

Index	1	2	3	4	5	6	7	8	9	...	25	26
LettersGuessedArray[2]										...		

For example:
`LettersGuessedArray[4]` stores an indicator that the *user* entered the letter `'D'`.

Suggestion 2
Use a one-dimensional array `LettersGuessedArray`, **Figure 5**, as follows:
Each letter entered is stored in this array at the next available cell.

For example:
`LettersGuessedArray[1]` stores the entered letter `'E'`,
`LettersGuessedArray[2]` stores the entered letter `'A'` because the *user* entered the letter `'E'` first followed by the letter `'A'`.

Figure 5

Index	1	2	3	4	5	6	7	8	9	...	25	26
LettersGuessedArray[2]	E	A								...		

An entered letter is never stored more than once.

[2] Your chosen programming language may use arrays with a lower bound value of 0.
If so, then assume that `LettersGuessedArray[0]` is not used.