# BCS THE CHARTERED INSTITUTE FOR IT

BCS HIGHER EDUCATION QUALIFICATIONS
BCS Level 4 Certificate in IT

## SOFTWARE DEVELOPMENT

Tuesday 27th September 2011 - Afternoon
Time: TWO hours

Section A and Section B each carry 50% of the marks. You are advised to spend about 1 hour on Section A (30 minutes per question) and 1 hour on Section B (12 minutes per question).

**Answer the Section A questions you attempt in Answer Book A**
**Answer the Section B questions you attempt in Answer Book B**

The marks given in brackets are **indicative** of the weight given to each part of the question.

Calculators are **NOT** allowed in this examination.

## SECTION A

Answer TWO questions out of FOUR in Answer Book A. Each question carries 30 marks

A1. The triangular table shown below shows a sequence of railway stations (such as A, B, C, D, E, F) on one line and the distances in kilometres (km) between all of them. For example,

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | 0 | | | | | |
| B | 5 | 0 | | | | |
| C | 15 | 10 | 0 | | | |
| D | 20 | 15 | 5 | 0 | | |
| E | 30 | 25 | 15 | 10 | 0 | |
| F | 35 | 30 | 20 | 15 | 5 | 0 |

In this example, the distance between stations E and C is 15km and between C and F is 20km.

a) Assuming a station name can be no longer than 40 characters, that each station has a unique 3 character code identifier and there are a maximum of 200 stations, define a suitable data structure to hold this information. The unique 3 character code identifier is not the same as any station name.

**(3 marks)**

b) Assume an integer array holds the distances between the first station and each of the other stations. In this example, it would be the distance between A & A, A & B, A & C, A & D, etc. and the first four values would be 0, 5, 15, and 20. Write a sub-program which

i) takes as its parameters the data structure you defined in a), an integer array holding the distances from the first station and each other station and an integer which gives the number of stations and
ii) prints out the triangular table of distances using the three character code identifier. (NOTE – You can assume the table will fit on the output medium. You do not have to worry about lines “wrapping around”.)

**(12 marks)**

Turn over]

c) Write an integer function which

   i) has four parameters; namely two parameters which can be either two station names, two unique 3 character codes or one station name and one unique 3 character code, one parameter representing the data structure you defined in part a) and an integer parameter which gives the number of stations and

   ii) returns the distance between the two stations passed to the function.

   If either of the station names or unique character codes does not exist, then the function returns a negative value.

**(15 marks)**

A2. A text file <intext> contains words separated by a space. The words contain only capital letters [‘A’..’Z’]. A sentence is terminated by a full stop.

The algorithm below is a first attempt to process the text to form an output file <outfile>. Letters are replaced by their lower-case counterparts, except the first one in any sentence. Other characters are copied to <outfile> unchanged.

```
Line                    code
  1    OPEN <intext> for reading, <outtext> for writing
  2    first_char ← TRUE
  3    WHILE NOT EOF(intext)  DO
  4        READ(intext, achar)
  5        IF achar contains a capital letter AND NOT first_char THEN
  6           BEGIN
  7              convert contents of achar to a lower-case letter
  8              first_char ← FALSE
  9           END;
 10        IF achar = stop  THEN first_char ← TRUE
 11        WRITE(outtext, achar)
 12    ENDWHILE
 13    WRITELN(“finished”)
```

a) Write out the code for the test of “achar contains a capital letter” on line 5 and “convert contents of achar to lower-case letter” on line 7.

**(5 marks)**

b) Dry run the code with the five character sequence *G. TH* read from <intext>. Note – In the sequence there is a space character between the full stop and the T.

**(20 marks)**

c) What conclusions do you draw from the dry run? Suggest ONE amendment.

**(5 marks)**

A3. The array **x** has been initialised as follows

| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| x | 1 | 9 | 5 | 0 | 8 | 6 | 3 | 5 | 1 | 0 | 2 | 9 |

The function ***a*** in the code below is going to be executed with parameter ***b*** set to 10 and parameter ***c*** set to 6. [*You can choose to follow either version of the code*]

a) Trace the call of the function ***a(10,6)*** and show clearly the results of the call.

**(8 marks)**

```
    Version 1 (C)            Version 2 (Pascal)
1   int a(int b, int c){     FUNCTION a(b, c : INTEGER):INTEGER;
2     int d,e;               VAR d, e : INTEGER;
3     /* begin function */   BEGIN
4     e = -1;                  e := -1;
5     d = 0;                   d := 0;
6     while(d < b)             WHILE d < b DO
7     {                        BEGIN
8       if( x[d] == c )          IF x[d] = c
9         e = d;                 THEN e := d
10      d++;                     d := d + 1;
11    }                        END;
12    return( e )              a := e
13  }                        END
```

b) Write a brief summary of what the function does.

**(6 marks)**

c) Decide on better names for the identifiers (the function name, its parameters and the variables) and rewrite the code [either version 1 or version 2] using your new names and including suitable comments.

**(10 marks)**

d) Rewrite lines 5 to 11 [of either version 1 or version 2] using a for-loop instead of a while-loop.

**(6 marks)**

A4. Base your answers on either program version 1 or version 2 below.

| Line No. | Version 1 | Version 2 |
|---|---|---|
| 1: | /* program compareTest */ | program orderTest ; |
| 2: | int v , w ; | var v , w : integer ; |
| 3: | int order( char c1, c2 ) | function order( c1, c2 : char ) : integer ; |
| 4: | int x ; | var x : integer ; |
| 5: | { | begin |
| 6: | if( c1 > c2 ) | if c1 > c2 then |
| 7: | x = 1 ; | x := 1 |
| 8: | else if( c1==c2 ) | else if c1=c2 then |
| 9: | x = 0 ; | x := 0 |
| 10: | else | else |
| 11: | x = - 1 ; | x := - 1 ; |
| 12: | return( x ) ; | order := x |
| 13: | } /* order */ | end /* order */ |
| 14: | void main( ) { | begin |
| 15: | w = 'a' ; | w := 'a' ; |
| 16: | v = order( w , 'A' ) ; | v := order( w , 'A' ) ; |
| 17: | } | end . |

| Key: description | | |
|---|---|---|
| A: integer constant<br>B: character constant<br>C: assignment operator<br>D: arithmetic operator<br>E: relational operator<br>F: result type | G: integer identifier<br>H: type identifier<br>I: function identifier<br>J: reserved word<br>K: comment<br>L: punctuation character | M: boolean (logical) expression<br>N: formal parameter<br>O: actual parameter<br>P: local variable<br>Q: global variable<br>R: assignment statement |

a) Copy and complete the following answer table. (You need only copy out either the “Text in version 1” or the “Text in version 2” column.) Then for each highlighted part in the programs above, decide what key it should have according to the table above.

**(18 x 1 marks)**

| Line | Text in version 1 | Text in version 2 | Key letters for highlighted code |
|---|---|---|---|
| 2 | v | v | |
| 2 | ; | ; | |
| 3 | int | integer | |
| 3 | order | order | |
| 3 | c2 | c2 | |
| 4 | int | integer | |
| 4 | x | x | |
| 6 | > | > | |
| 7 | = | := | |
| 7 | 1 | 1 | |
| 8 | c1 == c2 | c1 = c2 | |
| 9 | x = 0 | x := 0 | |
| 11 | - | - | |
| 10 | else | else | |
| 13 | /* ... */ | /* ... */ | |
| 15 | 'a' | 'a' | |
| 16 | v | v | |
| 16 | w | w | |

b) Find and copy out one example of each of the following

i) function call

ii) conditional statement

iii) function declaration

**(3 x 4 marks)**

## SECTION B

Answer FIVE questions out of EIGHT in Answer Book B. Each question carries 12 marks.

B5. a) Most programming languages provide a means of selecting one action from a group based on the value held in a variable. However, this value cannot be of type REAL. Explain why this is so.

**(3 marks)**

b) Develop an algorithm which reads a four-digit integer *Date* in the form mmdd (mm = month digits, dd = day digits) and assigns the appropriate season to a variable Time_of_Year. For example, if the *Date* was 1123 (that is, 23rd November) then “AUTUMN” would be assigned to Time_of_Year.

**(9 marks)**

| **Season** | **Dates** | | |
|---|---|---|---|
| SPRING | 21 March | to 20 June | inclusive |
| SUMMER | 21 June | to 20 September | inclusive |
| AUTUMN | 21 September | to 20 December | inclusive |
| WINTER | 21 December | to 20 March | inclusive |

B6. A number is perfect if the sum of its factors including 1 equals the number. Thus 6 is perfect as the factors of 6 (that is, 3, 2 and 1) when added together equal 6.

Write a program or pseudocode which prints out all the perfect numbers between 1 and an input limit *limit.*

**(12 marks)**

B7. The REAL function RANDOM() returns a value 0.0 < RANDOM() <=1.0.

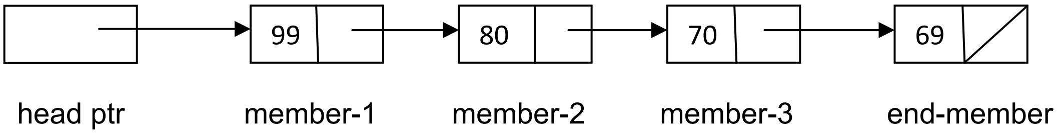
a) Write an expression which uses RANDOM() and gives an integer value between 1 and 6, as appropriate for a dice score.

**(3 marks)**

b) Incorporate this expression in pseudocode or a program which accepts an integer input to the variable *howmany*, and then carries out *howmany* simulations of a dice throw. These are stored in a suitable data structure. The mean value of all the throws is worked out and compared to the known value of 3.50. Separately add up all the scores for dice throw 1,2,…6.

**(9 marks)**

B8.



a) The diagram above represents a linked list with a single pointer field. Show the pointers if member-3 was removed from the list

**(2 marks)**

b) Write code to read in an integer value and delete the first member to contain this value. Note – The input value may not be held in the list.

**(10 marks)**

B9. One of the standard data structures used in programming is the *queue* with its distinctive operations of *join* (add a new value to the back of the *queue*) and *serve* (remove a value from the front of the *queue*). Using an array as the basic storage for the *queue*, write code in a language of your choice to implement the operations *join* and *serve*.

**(12 marks)**

B10. Give one reason why someone might prefer:

a) an interpreter *rather than* a compiler
b) a WIMP operating system *rather than* a command line operating system
c) an object-oriented programming language *rather than* a procedural language
d) a direct access file *rather than* a sequential access file
e) a high-level language *rather than* a low-level language
f) a linked list *rather than* an array

*[Note: It is expected that one well-chosen sentence per part will be sufficient as an answer]*

**(6 x 2 marks)**

B11. When designing a web page to take information from a user there are various form elements available. Name FOUR different form elements, giving an example of the kind of information that each is best suited for and drawing what they look like on screen.

**(4 x 3 marks)**

B12. Choose program version 1 or 2 below and then find occurrences of the following 6 errors. For each error you should give the line number and an explanation of the error. In addition you should state whether each error will be discovered at compile time or run-time.

a) identifier not declared
b) type error - index not allowed
c) array index out of bounds
d) syntax error
e) variable required
f) type error - invalid type

| line | Version 1 (C) | Version 2 (Pascal) |
|---|---|---|
| 1 | void main(); | PROGRAM p; |
| 2 | int x, y; | VAR x, y : integer; |
| 3 | char z[256]; | z : ARRAY[0..255] OF char; |
| 4 | { | BEGIN |
| 5 | w = 1; | w := 1; |
| 6 | x = 'w'; | x := 'w'; |
| 7 | 'w' = y; | 'w' := y; |
| 8 | x = z; | x := z; |
| 9 | if ( b > 1 b = b - 1 ; | IF b > 1 b := b - 1 ; |
| 10 | x[0] = y; | x[0] := y; |
| 11 | x = 0; | x := 0; |
| 12 | while( x < 9 ) { | WHILE x < 9 DO BEGIN |
| 13 | z[x-1] = x / y; | z[x-1] := x / y; |
| 14 | x++; | x := x + 1 |
| 15 | } | END |
| 16 | } | END. |

**(6 x 2 marks)**

** END OF EXAM **