BCS THE CHARTERED INSTITUTE FOR IT

BCS HIGHER EDUCATION QUALIFICATIONS
BCS Level 4 Certificate in IT

SOFTWARE DEVELOPMENT

Thursday 24th March 2016 - 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 <u>Section A</u> questions you attempt in <u>Answer Book A</u>
Answer the <u>Section B</u> questions you attempt in <u>Answer Book B</u>

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 2 questions (out of 4). Each question carries 30 marks.

General comments on candidates' performance

The standard of answers was generally high in this examination, however, for some candidates there remained areas where improvements could be made. The following comments should be noted:

- ***Answer all parts of the questions attempted.*** *Some candidates omitted parts of the question. Even if unable to provide a full answer, a partial answer might result in the extra marks that could make a difference between a fail and pass.*
- ***Avoid repetition.*** *Some candidates attempted to gain extra marks by repeating points already made. Future candidates are advised that no marks will be gained for such efforts.*
- ***Answer the questions set.*** *Some candidates attempted to gain marks by providing answers that were related only vaguely, if at all, to the questions set. Future candidates are advised that no marks are given for such answers.*
- ***Use care when writing code.*** *As mentioned previously, care and precision is necessary when answering questions using code. Often the examiners had difficulty identifying which variables or commands were being used; inevitably, the candidates lost marks as a result.*

Note that the answer pointers contained in this report are examples only. Full marks were given for valid alternative answers.

SECTION A

(Candidates were required to answer **TWO** out of the four questions set)

A1

In the array V below, the maximum value is 17 and the minimum value is 4 so the range (calculated as 17-4) is 13.

index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
V	10	9	17	7	12	8	5	15	4	6

a) Write a function minVal() to find the minimum value in an array V

(6 marks)

b) Similarly write a function maxVal() to find the maximum value in an array V

(6 marks)

c) Write a third function range1() to calculate the range of V (the difference between the max and min values) using minVal() and maxVal()

(6 marks)

d) Write a function range2() that does not call the function minVal() or maxVal() or any other functions written by you, but calculates the same result as range1().

(8 marks)

e) Choose which version of range calculation you prefer and state one reason why.

(4 marks)

Answer Pointers

a)
```
int minVal(){
        int i, max;
        min=V[0]; /* using index zero as initial value, means the loop can start at index 1 */
        for(i=1;i<=9;i++){
                if(V[i]<min){
                        min=V[i];
        }
        return min;
}
```
b)
```
int maxVal(){
        int i, max;
        max=V[0];
        for(i=1;i<=9;i++){
                if(V[i]>max){
                        max=V[i];
        }
        return max;
}
```

c)

```
int range1(){
        return maxVal()-minVal();
}
```

d)

```
int range2(){
        int i, max, min;
        max=min=V[0];
        for(i=1;i<=9;i++){
                if(V[i]<min){
                        min=V[i];
                if(V[i]>max){
                        max=V[i];
        }
        return max-min;
}
```

e)

Example:

Prefer range1() because it breaks down the problem into smaller, logical steps. The other functions might be more generally useful.

Example:

Prefer range2() because it is much more efficient for large arrays (instead of traversing the array twice, the answer is obtained in one traversal)

Examiners' comments:

The majority of candidates that attempted this question reached the required standard using solutions written in C, C++ or Java.

Parts a) and b) were generally well answered; however the solutions that gained full marks created efficient solutions by initialising max and min at the first element of the array V[0] and then comparing that value against the remaining elements using a FOR loop.

Most candidates gained full marks in part c) where the range was determined using the max and min function from parts a) and b). The majority of candidates determined the range in part d) using the approach outlined in the answer pointer above; however marks were also awarded in those case where candidates sorted the array and assigned the range to the difference between start and end elements of the newly sorted array.

Many candidates did not attempt part e); those that did gave well justified reasons for the method that they preferred.

A2

In a certain game played with cards, a player is dealt a hand of 7 cards.

Each card has a value (Ace(one), Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten, Jack(11), Queen(12) or King(13)) and each card has a Suit (Clubs(1), Diamonds(2), Hearts(3) or Spades(4)). So a card can be represented by 2 numbers (V,S). The first number V (in the range 1 to 13) represents the Value and the second number S (in the range 1 to 4) represents the Suit.

In the game it is important to know “the highest value of each Suit”.

An example of a hand of seven cards can be represented by two arrays

index	**1**	**2**	**3**	**4**	**5**	**6**	**7**
V	4	3	4	10	2	13	8
S	1	1	3	4	1	2	3

In this example of 7 cards, the highest value in Clubs is “Four” (i.e. S[1]=1, V[1]=4) and the highest value in Diamonds is “King” (i.e. S[6]=2, V[6]=13)..

You are required to write a program to find the highest value of each Suit.

Your program should assume the two arrays V and S exist and are already set up with the values to search. You should use this as data to discover the highest values.

The output from the program should be the four highest value cards (in this example, Four of Clubs, King of Diamonds, etc). Output “Not Found” for the value if a suit is not present in the hand.

(30 marks)

Answer Pointers

```
#include <stdio.h>
#include <string.h>
int V[]={0, 4, 3, 4,10, 2,13, 8}; /* Values */
int S[]={0, 2, 1, 3, 1, 1, 2, 3}; /* Suits */
int maxVal(int s){ /* what is the max value in suit s? */
        int max=0; /* 0 is equivalent to "not found" */
        int c;
        /*printf("maxVal %d\n",s);*/
        for(c=1;c<=7;c++){
          if(S[c]==s && V[c]>max) /* suit matches and value is bigger */
                   max=V[c];
        }
        return max;
}
void printCard(int v, int s){ /* convert number codes to printable strings */
        /*printf("printCard %d,%d\n",v,s);*/
        char valstr[10],suitstr[10];
        switch(v){
          case 1: strcpy(valstr,"Ace");break;
          case 2: strcpy(valstr,"Two");break;
          case 3: strcpy(valstr,"Three");break;
          case 4: strcpy(valstr,"Four");break;
          case 5: strcpy(valstr,"Five");break;
          case 6: strcpy(valstr,"Six");break;
          case 7: strcpy(valstr,"Seven");break;
```

```
            case 8: strcpy(valstr,"Eight");break;
            case 9: strcpy(valstr,"Nine");break;
            case 10: strcpy(valstr,"Ten");break;
            case 11: strcpy(valstr,"Jack");break;
            case 12: strcpy(valstr,"Queen");break;
            case 13: strcpy(valstr,"King");break;
            default: strcpy(valstr,"Not found");
        }
        switch(s){
            case 1: strcpy(suitstr,"Clubs");break;
            case 2: strcpy(suitstr,"Diamonds");break;
            case 3: strcpy(suitstr,"Hearts");break;
            case 4: strcpy(suitstr,"Spades");break;
        }
        printf("%s of %s\n",valstr,suitstr);
}
void findMaxVals(){ /* find the max val in each suit */
        int s,v=0;
        for(s=1;s<=4;s++){
            /*printf("findMaxVals %d,%d\n",v,s);*/
            v=maxVal(s);
            printCard(v,s);
        }
}
int main(){
        findMaxVals();
}
```

Examiners' comments:

This was the least popular questions in Section A and was only attempted by approximately 6% of candidates.

In a few cases the candidate produced a near correct structured solution based on determining the maximum numerical value of cards in each suit. Marks were lost when the output was printed in numerical format rather than using a switch statement to convert the numbers into text so for example (card = 12) and (suit = 0) should have been outputted as Queen of Clubs.

However few marks were gained by many candidates as the quality of their answers was generally poor; for example, many candidates got no further than simply using a single loop to find the maximum value of any of the cards in the hand.

A3

a) Trace the behaviour of the call ***h***(), when ***h*** and V are defined as shown below.

index	**0**	**1**	**2**	**3**	**4**	**5**
V	9	8	7	4	5	1

```
int f(g){if(g<0)return -g; else return g;}
int h(){
      int i=f(V[1]-V[0]);
      for(var j=2;j<6;j++){
            if(f(V[j]-V[j-1])>i)
                  i=f(V[j]-V[j-1]);
      }
      return i;
}
```

(14 marks)

b) Describe in a single sentence the overall effect of the function ***h***() **(6 marks)**

c) Rewrite the function ***h()*** changing all the identifiers to make the code more readable and understandable and add suitable comments. **(10 marks)**

Answer Pointers

a)

g=f(V[1]-V[0])=f(8-9)=f(-1)=1

h=2	f(7-8)>1	1>1	no	
h=3	f(4-7)>1	3>1,	yes,	g=3
h=4	f(5-4)>3	1>3	no	
h=5	f(1-5)>3	4>3	yes	g=4

return 4

b) function computes the largest gap (ignoring sign) between consecutive elements of the array

c)

```
int abs(v){if(v<0)return -v; else return v;}
int largestGapBetweenConsecutiveElements(){
      int gap=abs(V[1]-V[0]); /* between first two array elements */
      for(var i=2;i<6;i++){ /* now check all remaining pairs */
            if(abs(V[i]-V[i-1])>gap)
                  gap=abs(V[i]-V[i-1]); /* larger gap found */
      }
      return gap; /* return largest gap*/
}
```

Examiners' comments:

Many candidates made a good attempt at tracing the code in part a), although few actually completed the trace and stated that i=4 was returned.

Some candidates gained full marks in part b) for stating that the function returned the largest gap between array elements; but the majority believed it to be a sort or search algorithm. The task of making the code more readable in part c) gained little marks since few candidates realised they needed to change the identifiers to meaningful names and to add suitable comments to describe the process.

A4

a) Consider the code below and write it out in a more familiar human readable form.

```
void p(int q,char r){for(int s=0;s<q;s++)printf("%c",r);}p(10,'+');
```

(6 marks)

b) List the operators in the program above and, for each, state how many operands it has. One of the operators is a shorthand for something longer – write out the longer version.

(8 marks)

c) Referring to the code in part a), find and write out the following:

i) all the different identifiers
ii) all the different constants
iii) a conditional (logical, boolean) expression
iv) an iterative (repetitive, loop) statement
v) the statement that is repeated by the loop

(10 marks)

d) Write out the code from a) again, this time replacing the loop with an equivalent while loop.

(6 marks)

Answer Pointers

a)
```
void p(int q,char r){
        for(int s=0;s<q;s++)
                printf("%c",r);
}
p(10,'+');
```

b) operators

=, 2 operands	√√
<, 2 operands	√√
++, 1 operand, v++ is shorthand for v=v+1	√√ and √√

c)

i) identifiers p, q, r	√√√
ii) constants 0, "%c", '10', '+'	√√√√
iii) logical expression s<q	√
iv) iterative statement for(int s=0;s<q;s++)printf("%c",r);	√
v) repeated statement printf("%c",r);	√

d)
```
s=0;
while(s<q){
        printf("%c",r);
        s++;
}
```

Examiners' comments:

This was an extremely popular question that was answered by nearly all of the candidates, the majority of whom passed.

Part a) involved rewriting code in a human readable form was generally well answered; however maximum marks were only awarded where candidates made correct use of code indentation and spacing.

In part b) most candidates recognised the three operators (=, <, ++) but incorrectly listed the number of operands. Surprisingly many candidates were unaware that S++ was shorthand for S = S + 1.

Most candidates were able to rewrite the code using a while loop; however in some cases candidates lost marks by simply replacing for(int s=0, s<q, s++) with while(int s=0, s<q, s++).

SECTION B

Answer 5 questions (out of 8). Each question carries 12 marks.

B5

A business uses a Straight Line depreciation function to calculate the value of its computer system assets over their useful life. Straight Line depreciation assumes that an item will depreciate by a set amount each year, given by the formula:

$$\mathbf{D = (P - F) / t}$$

Where:

D	= Depreciation per year
P	= Purchase cost of the item
F	= Final value of the item
t	= Number of years the item will be in use

a) Write an expression in pseudocode (or a program in a language of your choice) in which variables P, F and t are input and the value of the item is printed each year after depreciation

(6 marks)

b) Using the pseudocode (or program) from part a), trace the value of the item where

Purchase cost = £40,000,
Final value F = £5,000 and
the number of years of the item t = 4.

(6 marks)

Answer Pointers

a) Accept pseudocode or program

```
Variable Declarations
          int t, i
          float Value, D, P, S
          INPUT “Number of Years:” t
          INPUT “Purchase Cost:” P
          INPUT “Salvage Value:” S
          BEGIN
                    i ← 1
                    D ← (P – S) / t;
                    Value ← P;
                    WHILE (i < t)
                              Value ← Value - D;
                              PRINT “Final Value = £ “Value;
                              PRINT “Year = “ i ;
                              Print newline
                              i++
                    END WHILE LOOP
          END
```

b) Trace table

Year (i)	D = (40K – 5K)/4	Value
0	8750	40,000
1	8750	31250
2	8750	22500
3	8750	13750
4	8750	5000

Examiners’ comments:

The majority of candidates who chose to answer this question elected to use a programming language rather than pseudocode. Marks were however lost if variables were not declared or if values were not assigned to these variables.

The question clearly requests a program which prints the value of the item ***each year*** *after depreciation. This requires an iteration which will repeat for the number of years specified by the variable t. Many candidates lost marks by providing a program which simply printed out the amount of annual depreciation.*

A number of candidates used the table structure set out in the answer pointers to answer part b). Other forms of answer were accepted if they clearly indicated the value of the variables. In general, answers to part a) were better than those to part b) and many candidates did not attempt part b) at all.

B6

Write a function ***factorial*** (n) in pseudocode (or a programming language of your choice) to calculate the factorial of n:

[Definition: factorial (n) = n * (n-1) * (n-2) ... * 2 * 1]

a) Using recursion **(6 marks)**

b) Using iteration **(6 marks)**

Answer Pointers

a) Recursion - accept pseudocode or program

```
FUNCTION Factorial (N: INTEGER) : INTEGER
    {Using Recursion}
    IF (N = 0)
        Factorial ← 1;
    ELSE
        Factorial ← N * Factorial (N-1);
    END
END FUNCTION
```

b) Iteration - accept pseudocode or program

```
FUNCTION Factorial (N: INTEGER) : INTEGER
    {Using Iteration}
    INTEGER Temp, X
    Temp ← 1;
    FOR (X=1; X<=N; X++)
        Temp ← Temp * X;
    END
    Factorial ← Temp;
END FUNCTION
```

Examiners' comments:

This was not a popular question and when it was attempted candidates either gained most of the marks available or else provided solutions that did not address the question at all. Some candidates were only able to provide solutions to part b) indicating that they had not encountered recursion as part of their studies.

Some candidates were unable to work from the definition of factorial given in the question to derive a form that could be incorporated into code or pseudocode. In these answers the definition was copied as-is into a code or pseudocode statement. Answers of this form were marked down.

As is to be expected, some candidates were able to provide solutions which dealt with the main calculation but did not correctly deal with loop termination conditions. Partial credit was awarded to these solutions.

B7

a) Explain the difference between a compiler and interpreter.

(4 marks)

b) Discuss the advantages and disadvantages between using a compiler and interpreter.

(8 marks)

Answer Pointers

a) A compiler converts the source code to object code, which the computer can run. The object code is converted to match the target computer that will run the software.

Interpreter software executes the source code directly; the code is translated one line at a time which avoids the need to compile the program.

b) Discussion points could include:

- Interpreted software runs more slowly than compiled software as each statement in the program is analysed before it is translated and executed.
- Compiled are more suitable for large projects as the code is translated faster.
- The source code is more secure when using a compiler as it does not need to be supplied to the client;
- The source code is supplied to the client when translating using an interpreter.
- The compiled code may be optimized to run faster on a target machine
- Interpreted code can be run on any machine that has an interpreter installed on it
- Errors in code that are detected with an interpreter can be edited without translating all of the code again
- A compiler will create an error listing during the compilation process; these errors need to be corrected and all of the code needs to be translated again

Examiners' comments:

Credit awarded to answers to this question was largely influenced by the fact that whilst candidates in general had a good grasp of the purpose of a compiler (except for those who believed that the primary purpose of compilation is to detect errors in programs!) they were uncertain about the purpose of an interpreter. The most common answer to part a) was "A compiler translates all the source code to machine code. An interpreter translates source code to machine code a line at a time.". This answer captures the fact that interpreters work on a line at a time basis but fails to identify the primary characteristic of an interpreter; i.e. it executes code. Clearly interpreters will undertake some form of input parsing (not necessarily to machine code) but their primary role is to perform the instructions they are given.

Given this uncertainty as to the purpose of an interpreter, some of the answers to part b) were poorly informed. Where candidates were aware of the primary difference between a compiler and an interpreter there was often a confusion between the process of compiling and the execution of compiled code. Some candidates simply wrote down any possible difference that occurred to them regardless of whether these could be justified or not.

B8

a) Using an algorithm, describe the operation of a stack. **(9 marks)**

b) A stack contains the values 7, 6, 9 where 7 is the first value stored and 9 is the last. List the final values in the stack after the following operations are made:

1. Pop
2. Pop
3. Push12
4. Push16
5. Pop
6. Push 11

(3 marks)

Answer Pointers

a) A stack is used to as a temporary storage space for data. It is defined as a linear data structure which operates on a **last in, first out (LIFO)** basis. The stack uses a single pointer (index) to keep track of the data in the stack, where:

- Data is inserted (or pushed) onto the stack and
- Data is removed (or popped) from the stack.

Initially the Stack-pointer is set to 1 and Stack has “Max” elements

Stack Push Algorithm (data inserted onto stack)

```
if stack_pointer >= MAX then
    return stack full message
else begin
    stack[stack_pointer] = data;
    stack_pointer = stack_pointer + 1;
end
```

Stack Pop Algorithm (data removed from stack)

```
if stack_pointer <= 1 then
    return stack empty message
else begin
    stack_pointer = stack_pointer - 1;
    data = stack[stack_pointer];
end
```

b) Final content of stack 7, 12, 11

Examiners’ comments:

A third of the marks for part a) were available for candidates who knew that a stack was a LIFO structure which supported the operations push and pop. Quite a few candidates stated that a stack was a FILO (First In Last Out) structure. Although this term is not normally used in text books, full credit was given to it. Candidates who attempted part a) were generally able to give acceptable algorithms for push and pop although some did not deal correctly with the stack full and stack empty conditions.

Candidates generally answered part b) by writing down three numbers. If they were the correct three numbers in the correct order then full credit was rewarded. Partial credit could only be awarded for incorrect answers if candidates had shown how they arrived at their answer. As with questions testing numeracy it is possible to gain marks for showing a knowledge of process even if that process is inaccurately applied.

B9

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)

Answer Pointers

Form Element	**Example Usage**	**Drawings**
Single-line text Multi-line text	Single text – Name, email, mobile Multi – Address	Accept any reasonable attempt to draw the form element as on a web browser display.
Menu Selection	Choose One from many options or MANY from many options – eg. select country	
Radio Buttons	Choose ONE from a range of options	
Check Boxes	Choose MANY from a range of options	
Submit Buttons	Send completed form to server	

Examiners' comments:

This was a popular question which in general was well answered. A large number of candidates gained the full twelve marks. A very few answers discussed the general principles of web page design but as the question specifically asked for a discussion of form elements these did not receive any credit.

B10

Compare the following pairs of terms.

a) Truncation and rounding **(4 marks)**
b) Static and dynamic data structures **(4 marks)**
c) Flat file and relational databases **(4 marks)**

Answer Pointers

a) Rounding is to replace a number with an approximate value using fewer digits, so 34.5674 can be rounded to 34.57 which is rounded to 2 decimal places. Truncation is the process of limiting the number of digits to the right of the decimal point. So 34.5674 can be truncated to 34.56, which is truncated to 2 decimal places.

b) Data structures can be termed static, this is where the data structure size is fixed; a typical example is where a list of data is declared as an array in a computer program. Alternatively data structures need to be dynamic where the data storage requirement is not known before program execution; dynamic data structures are used where the amount of data stored can be increased or decreased as needed.

c) A flat file database is a single table in which each row of the table is known as a record and each column is known as a field, whereas a relational database comprises of more than one table where tables that share a relationship are joined by linking a primary key in one table to a foreign key in another table.

Examiners' comments:

There were a large number of partial answers to this question where candidates were confident that they could answer one or two of the parts but not all the question.

There were surprisingly few fully correct answers to part a). In general candidates were aware of what rounding is but unable to describe truncation. The most succinct way to answer the question was to give an example of rounding and truncating an actual number e.g. 12.99.

In part b) candidates were able to guess that static data structures would not change whereas dynamic structure would change. There were not however able to identify what aspects of the data structures altered. Many suggested that the contents of a static structure were constant whereas they could be changed in a dynamic structure.

In answers to part c) candidates seemed to be unaware of the term 'flat file'. The majority of credit was therefore gained by describing the characteristics of a relational database. Such descriptions were often overlong given the number of marks available. The important distinction is that in a flat file the structure of the data is defined internally by programs that access it, whilst in a relational database the structure of the data is defined in the database.

B11

a) Name the stage of the traditional software development process in which an algorithm would be created and the tasks it contains.

(4 marks)

b) Name the stages that precede and follow the stage mentioned in a), describing the links between these three stages with reference to the algorithm.

(8 marks)

Answer Pointers

a) Stage: Design
Write a programming description of how a task can be achieved.

b) Stages: Specification > Design > Implementation

1. Take the requirement as described in the specification
2. Then design a programming description of how it can be achieved
3. Finally implement by coding into a particular programming language.

Examiners' comments:

Part a) of the question required candidates to name the stage in the software development life cycle (SDLC) in which an algorithm would be created. A number of candidates recognised that the SDLC was the subject of the question and described every stage in the life cycle. This did not obtain credit unless as part of the description Design was identified as being specifically relevant to the question.

In part b) candidates were asked to discuss three stages of life cycle: Design, the stage following Design (Implementation) and the stage preceding Design (Specification). Many candidates chose to discuss all the stages following Design and omitted a discussion of Specification.

B12

Consider the following function called ***findmax*** together with a single test case as shown below:

```
int findmax(int low, high){
/* specification: find the maximum value in array v between index low and index high inclusive */
int max; /* holds the maximum value found so far */
 int i; /* loop counter */
max=100;
for ( i=low; i<high; i++)
        if( v[i] > max )
                max=v[i];
return(max);
}

test case: v[1]=55; v[2]=4; v[3]=16; low=1; high=3;
```

a) How would a black-box test of the function ***findmax*** be performed?

(3 marks)

b) In what way would the error in the function ***findmax*** show up under black-box testing?

(3 marks)

c) How would a white-box test of the function ***findmax*** be performed?

(3 marks)

d) In what way would the error in the function ***findmax*** show up under white-box testing?

(3 marks)

Answer Pointers

a) In black box testing the function *findmax* is available as an executable and it can be run to see if the results obtained match the expected results based on the specification.

b) Using black-box testing the result of running *findmax* with the test case data is 100. This is incorrect as the expected result is 55 based on the test data specified.

c) In white-box testing the source code of *findmax* is available and so the code can be dry run line-by-line to look for errors.

d) Using white-box testing the code would be dry run tested using the data specified. The result would have been 100 rather than the expected result of 55, with the error caused by the assignment "max =100".

Examiners' comments:

This question invited candidates to explain the terms 'black box testing' and 'white box testing'. They were able to gain further marks by applying their knowledge of these forms of testing to the code set out in the question.

In general, most candidates attempting this question were able to give satisfactory definitions of the two types of testing. That is to say, most marks were gained by answers to parts a) and c) of the question.

Candidates were less successful in describing how the testing techniques would be applied to the code. Many thought that the result of testing would be a syntax error indicated by a compiler! The best answers identified the problem with the code that was given and were aware that black box testing would indicate that an error was present whereas white box testing could show where the error was located.