THE BCS PROFESSIONAL EXAMINATION

BCS Level 4 Certificate in IT

September 2017

EXAMINERS' REPORT

SOFTWARE DEVELOPMENT

General comments on candidates' performance

The standard for this examination was reasonable, with many candidates gaining high marks.

However, although there were some excellent papers, a large number of candidates lost marks by providing answers that contained minimal content or were unrelated to the question. Candidates are therefore advised to spend more time reading and understanding the requirement of each question before writing their answers. In addition, many candidates answered more than the required number of questions. Future candidates should note that no credit is given for answering additional questions.

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

SECTION A

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

A1

Two measuring devices, at opposite ends of a room, are measuring the room temperature and readings are taken every hour.

a) Write a function (called highDevice) to create a new array (called HIGH) which has an entry 1 if DEVICE1 is reading higher than DEVICE2, 0 if they are the same and 2 if DEVICE2 is reading higher than DEVICE1.

(**8 marks**)

```
DEVICE1 18 19 20 21 22 21 20 19 18 17 16 18
DEVICE2 18 19 21 21 20 19 20 19 19 16 17 18

HIGH     0  0  2  0  1  1  0  0  2  1  2  0
```

b) Write a function (called doCount) to find out how many times a particular number (supplied as a parameter) appears in an array (in the example above the number 2 appears 3 times in the array HIGH).

(**8 marks**)

c) Write a function (called doRMS) to find the root-mean-square difference between the readings of DEVICE1 and DEVICE2 (that means: find all the differences in the readings, square them, find the average and then take the square root of the average).

(**8 marks**)

d) Write a program, using the functions in parts a), b) & c) to report the RMS difference and to report which of the two devices reads high most often (or whether it is a tie).

(**6 marks**)

Answer pointers

```
#include <stdio.h>
```

```
#include <math.h>
int DEVICE1[12]={18,19,20,21,22,21,20,19,18,17,16,18};
int DEVICE2[12]={18,19,21,21,20,19,20,19,19,16,17,18};
int HIGH[12];
int max=12; //should be 24 (hours in a day)
void highDevice(){
	int r;
	for(r=0;r<max;r++){ //r for reading
		if(DEVICE1[r]>DEVICE2[r]) HIGH[r]=1;
		else if(DEVICE1[r]==DEVICE2[r]) HIGH[r]=0;
		else  HIGH[r]=2;
	}
}
int doCount(int n){
	int count=0;
	int i;
	for(i=0;i<max;i++){
		if(HIGH[i]==n)
			count++;
	}
	return count;
}
float doRMS(){
	int total=0;
	int i;
	for(i=0;i<max;i++){
		float dif=DEVICE1[i]-DEVICE2[i];
		total+=dif*dif;
	}
	return sqrt(total/max);
}
int main(){
	highDevice();
	float rms=doRMS();
	printf("RMS of readings for DEVICE 1 and DEVICE2 %f\n",rms);
	int count1=doCount(1);
	int count2=doCount(2);
	if(count1>count2)
		printf("DEVICE1 reads high most often");
	else if(count1==count2)
		printf("DEVICE1 & DEVICE2 tie");
	else
		printf("DEVICE2 reads high most often");
}
```

Examiners' Guidance Notes

There is evidence that many of the candidates that attempted the question produced a correct or near correct solution for part (a) where a function (HighDevice) was created to store the highest reading from the two temperature devices at opposite ends of a room in a HIGH array. In part (b) functions were used to determine the number of times that a number (number input as a function parameter) appeared in the HIGH array and in part (c) functions were used to determine the RMS difference between the readings for the temperature devices.

The majority of candidates lost marks where they did not attempt part (d) or failed to include the correct code to print out which of the temperature devices reads high most often.

A2

All the students taking a module have filled in a questionnaire which contained 10 questions. Each question was answered with the same scale:

Strongly Disagree [ ] Disagree [ ] Neutral [ ] Agree [ ] Strongly Agree [ ]

The students answers have been coded from 1 (Strongly Disagree) to 5 (Strongly Agree) so that the 10 answers from each student are coded as 10 numbers from 1 to 5. This data is now ready to be analysed.

You are to write a program to read and analyse the data from 50 students. The students' answers are to be given scores [-5, -2, 0, 2, 5] so that "Strongly Disagree" is scored -5 and "Strongly Agree" is scored 5.

The average score for each question should be calculated and reported. Finally, the question with the lowest total score should be reported.

(**30 marks**)

[Note: For maximum marks your program should make use of functions and it should be possible to change the scoring system easily (e.g. from [-5, -2, 0, 2, 5] to [-2, -1, 0, 1, 2]).]

Answer pointers

Use of functions (e.g. readAndTotal, findMin) 10 marks (5 marks each)
Scoring system easily changed (use of array?) 5 marks
Note: This answer does NOT use functions

```
#include <stdio.h>
int main(){
int SCORE[6]={0,-5,-2,0,2,5}; //set up an array of scores (will use indexes 1..5)
int TOTAL[10];
int s, q, answer, low;
for(q=1;q<=10;q++){ //q for question
      TOTAL[q]=0; //initialise TOTAL array
}
for(s=1;s<=50;s++){ //s for student
      for(q=1;q<=10;q++){ //q for question
            scanf("%d", &answer);
            TOTAL[q]+=SCORE[answer];
      }
}
for(q=1;q<=10;q++){ //q for question
      printf("%.2f ",TOTAL[q]/50.0); // print average to 2 dec.pl.
}
low=1; //start by assuming that the lowest score is in position 1
for(q=2;q<=10;q++){ //q for question, look through rest of questions
      if(TOTAL[q] < TOTAL[low]){ //we have a new low
            low=q;
      }
}
printf("\nThe lowest score is for question %d.",low);
}
```

Examiners' Guidance Notes

This was the least popular question in Section A. In some cases, candidates produced a structured solution to gain a satisfactory pass mark.

However, the evidence shows that the majority of candidates seemed to misunderstand the requirements of the question; consequently, few marks were gained by the majority of candidates as the quality of their answers was generally weak; for example, many candidates simply entered a list of assignments for the scoring system numbers and didn't consider using a For Loop to enter the answer data for the 50 candidates.

A3

a) Give the final values of the variables a, b, c, d after the following code has executed:

(**4 marks**)

```
i=2; a=--i; b=i;
i=4; c=i--; d=i;
```

b) Based on your answer to part a), or otherwise, state the key difference between --i and i--.

(**4 marks**)

c) Given the initial values in the array V as follows:

```
index 0   1   2   3   4   5
V     17  15  13  12  14  16
```

trace the execution of the function call ***f***(5), where the function ***f*** is defined as follows:

(**18 marks**)

```
int f(int g){
      int h=V[g];
      int i=g;
      while(i>0){
            int j=V[--i];
            if(j<h)h=j;
      }
      return h;
}
```

d) Describe in your own words the result that function ***f*** produces and thus give it a more meaningful name.

(**4 marks**)

Answer pointers

a) a=1, b=1, c=4, d=3

b) both --i and i-- have the effect of decrementing the value of i by 1, but when --i occurs inside a larger expression the value delivered is the decremented value, whereas the delivered value of i-- is the original value of i (before decrementing).

c) trace:

```
g    h    i   i>0   j    j<h
5
     16   5                      h=V[g];i=g
               T                 while(i>0
          4         14   14<16    j=V[--i];if(j<h)
     14                               h=j
               T                 while(i>0
          3         12   12<14    j=V[--i];if(j<h)
     12                           h=j
               T                 while(i>0
          2         13   13<12    j=V[--i];if(j<h)
               T                 while(i>0
          1         15   15<12    j=V[--i];if(j<h)
               T                 while(i>0
          0         17   17<12    j=V[--i];if(j<h)
               F                 while(i>0
return 12
```

d) The function f starts at the end (or rather the index g) of array V and travels backwards through the array recording the smallest element found so far in h. Therefore, the function returns the smallest element found in V. So, f should be called "smallest" or "min" or "least", etc.

Examiners' Guidance Notes

This was a popular question attempted by 86% of the candidates, where several candidates gained maximum or near maximum marks for a correct solution to all parts.

However only 32% of candidates achieved a satisfactory pass mark, with evidence showing the main problem areas being:

- *Parts (a and b) – mistakes were made as candidates were confused by the differences between --i and i--*
- *Parts (c and d) – there were frequent errors in tracing the function f(5), consequently the candidates were unaware that the trace returned the smallest member of the array V.*

A4

Based on the following program extract, answer the questions below:

```
#include <stdio.h>
float g(int x, int y, char a){
      float r;
      if(x<0||y<0)return -1;
      if(a=='*'){r=x*y; return r;}
      if(a!='/')printf("unexpected:%c",a);
      return(x/y);
}
```

a) List all the identifiers in the extract.

(**5 marks**)

b) List all the operators.

(**5 marks**)

c) List all the constants.

(**5 marks**)

d) List the types of each of the constants in c).

(**5 marks**)

e) From the program extract, copy out an example of each of the following:

a declaration,
a boolean expression,
an assignment statement,
a conditional statement.

(**10 marks**)

Answer pointers

a) identifiers**:** **g, x, y, a, r**

b) operators: **<, ||, <, ==, =, *, !=, /**

c) constants: **0, 0, -1, '*', '/', "unexpected:%c"**

d) types: **int, int, int, char, char, string**

e) a declaration: **float r**

a Boolean expression: **x<0**

an assignment statement: **r=x*y;**

a conditional statement: **if(x<0||y<0)return -1;**

Examiners' Guidance Notes

This was an extremely popular question that was attempted by 98% of the candidates; it was generally well answered with approximately 74% of the candidates achieving a satisfactory pass mark.

Most of the candidates were able to gain maximum or near maximum marks for parts (a, b). Marks were mainly lost in part (c, d) where evidence shows that many candidates failed to identify all of the constants and the types of constant used in the code. In part (e), candidates often made further errors in identifying Boolean expressions and conditional statements correctly.

SECTION B

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

B5: Values for the hyperbolic cosine function are obtained from the power series

$Cosh(x) = 1 + x^2 / fac(2) + x^4 / fac(4) + x^6 / fac(6) + \ldots$

Where: fac(n) = factorial n = 1*2*3*4*…n

Note: - Use pseudocode or actual program code of your choice to answer this question.

a) Write code for fac(n); any method may be used.

(4 marks)

b) Incorporate your function into another function HCos(x) which calculates Cosh(x) using the power series given earlier. Show how to terminate the calculation when the difference between successive terms is less than 0.00005.

(8 marks)

Answer Pointers

a) An iterative or recursive method for factorial code were both acceptable

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

b)

```
FUNCTION HCos(x)

        Accuracy ← 0.00005;   // Initialisation
        Sum ← 1;
        N ← 0;
        Multiplier ← 1;
        Past Term ← 1;
        Difference ←1;

        WHILE (Difference > Accuracy) DO
                Multiplier ← Multiplier * x * x;
                N ← N + 2;
                Term ← Multiplier / Factorial(N);
                Sum ← Sum + Term;
                Difference ← Abs(Past Term - Term);
                Past Term ← Term;
        END WHILE

        HCos ← Sum;

END FUNCTION
```

Examiners' Guidance Notes

This was the least popular question attempted in Section B by only 14% of the candidates. Overall performance on this question was poor with just over a third achieving a satisfactory pass mark. However there was a slight improvement in performance compared with the equivalent question on last years' paper. Only a few candidates produced a correct or near correct solution.

Breaking the question down to its two parts the evidence shows that:-

- *Part a) was attempted using recursion by most candidates, producing generally correct algorithms and/or code.*
- *Part b) candidates generally produced incorrect algorithms and did not express the formula in code/pseudocode. There were many examples of poor understanding of how an existing function (e.g. fact) is called within the main body of another function.*

B6: Iteration structures are commonly used in computer programming. Describe, with examples **THREE** iterative techniques.

(3 x 4 marks = 12 marks)

Answer pointers

Iteration Technique	**Examples in C**

for loop
The for loop control method is useful when the same instructions or calculations have to be carried out for a known number of iterations.
The "for" statement consists of the initial expression, the conditional expression and the loop counter.
	// for loop example
	int x;
	int total = 0;
	for (x = 0; x < 10; x++)
	{
	total = total + x;
	}

while loop
In the while control method technique, the loop operates when a condition is satisfied at the start of the loop and stops when this is no longer true.
	// while example
	int x = 0;
	int total = 0;
	while (x < 10)
	{
	total = total + x;
	x++
	}

do-while loop
In the do-while control method technique, the loop operates at least once, as the condition isn't checked until the end of the block.
The logical condition is tested at the end of block and repeats until the condition is no longer true.
	// do_while example
	int x = 0;
	int total = 0;
	do
	{
	total = total + x;
	x++
	} while (x < 10);

Examples in other programming languages were acceptable

Examiners' Guidance Notes

This was a fairly popular question attempted by 40% of candidates with more than 60% achieving a satisfactory pass mark, the best overall performance from Section B.
Many candidates produced good illustrative examples of code and were awarded high marks. However, there is evidence that some candidates failed to identify the DO/WHILE loop as an alternative to the WHILE loop.
There were a small number of candidates who may have misread the question. Those candidates interpreted iteration in many different contexts (such as the SDLC) and therefore did not score any marks.

B7:

a) Explain the difference between internal and external sorting.

(4 marks)

b) Describe the stages that take place when performing a merge sort.

(6 marks)

c) State one advantage and one disadvantage of using the merge sort procedure.

(2 marks)

Answer Pointers

a) In internal sorting, all the data to sort is stored in memory at all times while sorting is in progress. In external sorting, data is stored outside memory (for example on a tape drive) and only loaded into memory in small amounts; external sorting is usually applied in cases when all the data can't fit into memory entirely.
b) Merge sort is an efficient, general purpose sorting algorithm that is suitable to re-arrange large sets of data in lists or to sort sequential data from file streams. It performs the following stages:
    - Divide the unsorted lists into a series of n sub-lists, where each sub-list contains one element.
    - Each of the sub-lists is therefore sorted as it contains only one element.
    - The sub-lists are repeatedly merged to create new sorted sub-lists.
    - This continues until all sub-lists have been merged into one sorted list.
c) Merge sort is a fast technique that is useful in sorting sequential data; however, it uses a large amount of space which can be problematical with large data sets.

Examiners' Guidance Notes
This was the second least popular question attempted in Section B by just over a quarter of candidates. Overall performance on this question was very poor, the worst in Section B, with less than 20% achieving a satisfactory pass mark. The average mark was particularly low and seems to indicate candidates had not adequately revised this topic because they tackled the question showing very little knowledge. For example, candidates were unfamiliar with the merge sort and produced vague irrelevant attempts at anything they could recall about sorting techniques.

B8:

a) What is normally meant by the term 'debugging'?

(4 marks)

b) How is debugging approached in a simple programming environment where the programmer has only the standard output facilities of the programming language to use for this purpose?

(4 marks)

c) What extra facilities to assist in debugging might be provided in a more extensive development environment?

(4 marks)

Answer Pointers

a) The program does not respond in the way that was expected and the programmer is looking for more diagnostic information to discover the source of the problem and fix it. Debugging is the stage after checking that the syntax is correct.

b) The programmer can insert extra output statements in key places in the code. It is usual to output a distinctive label to show that this place has been reached at run-time and often it is useful to output the value of some key variables. Key places in the code are just inside loops (to find out how many times they are repeating) and on the separate branches of conditionals (to understand which branch was taken).

c) In a more sophisticated development environment, the programmer might expect to be able to set 'breakpoints' where the program can be temporarily suspended and inspections made of the values of variables, etc.

Examiners' Guidance Notes

This was a fairly popular question attempted by almost three quarters of all candidates, with just over a third achieving a satisfactory pass mark. This was surprising given the popularity of the question.

Breaking the question down to its three parts the evidence shows that: -

- *Part a) was reasonably well answered but candidates would benefit from elaborating on their answers to get full marks. Many candidate answers were too brief or too vague and verbose.*
- *Part b) candidates generally produced a wide range of answers covering knowledge of debugging techniques. Many candidates diverted from the main theme of the question and focused unnecessarily on testing.*
- *Part c) Very weak answers with quite a few cases of candidates misunderstanding what was required. However, credit was given to a range of answers covering IDE frameworks and the tools they support but again the description of how these tools work generally lacked any significant detail.*

B9: Briefly describe the type of testing you would use for:

a) Unit tests

(6 marks)

b) Integration testing

(6 marks)

Give reasons for your answers.

Answer pointers

The primary goal of unit testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and determine whether it behaves exactly as you expect. Each unit is tested separately before integrating them into modules to test the interfaces between modules. Unit testing has proven its value in that a large percentage of defects are identified during its use. White box logic tests would be suitable for unit testing.

Integration testing is a logical extension of unit testing. In its simplest form, two units that have already been tested are combined into a component and the interface between them is tested. A component, in this sense, refers to an integrated aggregate of more than one unit. In a realistic scenario, many units are combined into components, which are in turn aggregated into even larger parts of the program. The idea

is to test combinations of pieces and eventually expand the process to test your modules with those of other groups. Eventually all the modules making up a process are tested together. Beyond that, if the program is composed of more than one process, they should be tested in pairs rather than all at once.

Integration testing identifies problems that occur when units are combined. By using a test plan that requires you to test each unit and ensure the viability of each before combining units, you know that any errors discovered when combining units are likely related to the interface between units. This method reduces the number of possibilities to a far simpler level of analysis. Black box testing would be appropriate for integration testing.

Examiners' Guidance Notes

This was a fairly popular question attempted by just over three quarters of all candidates, 44% achieving a satisfactory pass mark. The evidence shows that many candidates could differentiate unit and integral testing but lacked knowledge of the actual strategies used. The emphasis (for example) of using white box testing for unit testing was rarely fully addressed. The process and staging of these testing strategies were either ignored or in many cases produced vague and woolly answers.

B10. Write notes to compare and contrast the following pairs of terms

a) source code and object code
b) linkers and loaders
c) multi-user and multi-tasking

(3 x 4 marks = 12 marks)

Answer Pointers

a) **Source code** is the language instructions that have been written by the computer programmer; the computer cannot run or execute the source code directly. **Object Code** (or executable code) is translated from the source code using an assembler or compiler into numerical instructions that can be understood and executed by the computer.

b) **Linkers** are computer programs that are used to link one or more link object code files into a single executable program. Computer programs usually consist of several modules each of which has been compiled; these modules need to be linked together before the main program can run. **Loaders** are computer programs that insert the machine code produced by the assembler into the correct location in the computer's main memory.

c) **Multi-tasking** is when a user has more than one program open (or loaded into RAM) at the same time. The operating system allows the user to switch between programs with each program running in a different window. **Multi-user** operating systems support a network where more than one user can access the same data at the same time; it appears to each user as if they are the only user on the system. In the multi-user environment the operating system will control access by the use of usernames and passwords.

Examiners' Guidance Notes

This was the most popular question in Section B attempted by 86% of all candidates,
The overall performance was good with about 58% achieving a satisfactory pass mark.
Breaking the question down to its three parts the evidence shows that: -

- *Part a) was reasonably well answered but candidates should expect to elaborate on their answers to get full marks. There were some candidates who thought, incorrectly, that object code related to object oriented programming and misread the question and its context/topic area.*

- *Part b) many candidates failed to produce satisfactory knowledge of linkers and loaders in particular. A wide range of interpretations were made by the weaker candidates who clearly misunderstood the topic area covered in this question.*
- *Part c) there were many good attempts with most candidates offering satisfactory knowledge distinguishing these modes of processing.*

B11. Explain the difference between open source and closed source software. Compare and contrast these two approaches to software provision.

(12 marks)

Answer pointers

Open source software is freely available for download from the Internet. The user is permitted to copy, edit and change the software as required.

Benefits include: These software applications can be downloaded for free from the Internet and have similar features to their rival closed source software and the code for the software is available so it can be modified to solve a specific problem. **Limitations** include: open source packages generally lack the dedicated support of closed source software and software updates are usually unscheduled and sporadic. **(6 marks)**

Closed source software is licensed to the purchaser and gives the right to use the software, but not to modify it or redistribute it. The software developer holds the copyright to the software product.

Benefits include: the manufacturer normally provides support for their package, such as: help pages, wizards and online video instructions and the software manufacturer tests the system extensively before release to ensure the software is robust. **Limitations** include: the software may have many complex features that are paid for but never used and the software is often large and complicated and takes a considerable time to learn. (**6 marks)**

Examiners' Guidance Notes

This was the second most popular question in Section B attempted by 82% of all candidates,
The overall performance was good with about 58% achieving a satisfactory pass mark.
It was pleasing to see a number of candidates producing correct or near correct solutions, with both parts equally well answered by most candidates. There is evidence to show the only main problem area was that many candidates wrote a lot of memorised material on these topics and that failed to answer the question directly. Candidates should be aware that marks were awarded for answers that discussed the trade-offs with a satisfactory explanation of the benefits and drawbacks when comparing each approach.

B12: Documentation is an important aspect of software development, write notes on the following:

a) The purpose, the content and the methods for delivering end-user documentation.

(6 marks)

b) The content and purpose of the requirements documentation.

(6 marks)

Answer Pointers

a) End-user documentation is designed to assist and support users in using a software product or service. Typical user documentation would describe the various features in a software application and the methods available to operate these features effectively.
Traditionally this documentation consisted of an indexed user manual with notes and screen shots on the features of the program and trouble-shooting methods.

Currently user documentation might include the following; FAQs, online support and video tutorials. Many software applications include embedded on-screen support, such as: tool tips and dynamic page content (wizards).

b) The requirements for a software development are created and documented during the analysis phase of the SDLC. The requirements are a description of the functionality of a software application and are used throughout the software development to explain the intended operation of the software. The exact content of this documentation is dependent upon the application being developed but may include the following: the exact functionality of the software, the interaction of the software with other programs or users, response time, maintainability and security features.

Examiners' Guidance Notes

This was a popular question attempted by 76% of all candidates.

The overall performance was good with about 56% achieving a satisfactory pass mark.

Candidates produced a wide range of answers in terms of quality and quantity. There is evidence that the distinction between the two types of documentation was not always clear and adequately explained and marks were lost as a result. The best answers were from those candidates who could relate their answers to their own practical experience and knowledge.