THE BCS PROFESSIONAL EXAMINATION

Level 4 Certificate in IT

March 2015

EXAMINERS' REPORT

SOFTWARE DEVELOPMENT

General comments on candidates' performance

The standard of answers was generally higher than in previous examinations. However, for some candidates there remained areas where improvements could be made. The following comments should be noted:

- ***Understand the requirements of the question.*** *Candidates are advised to spend time reading and understanding exactly what is required, both in terms of the number of questions to answer and the precise requirement of each question.*

- ***Avoid repetition.*** *Some candidates attempted to gain extra marks by repeating points already made. Candidates are advised that no extra marks will be gained by repeating points already made.*

- ***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 failure and pass.*

- ***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. Candidates are advised that no marks are given for answers that do not address the question asked.*

- ***Use care when writing code.*** *Care and precision are 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.*

Please 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

{ please see exam paper for question }

Answer pointers

a)

```
#include<stdio.h>
int main(){
        int A[100];
        /* not interested in how A gets filled */
        int S=10; /* not interested in where this number comes from */
        int i;
        int found=0; /*using 0 for false, 1 for true */
```

```
	for(i=0;i<100;i++)
		if(A[i]==S)
			found=1;
	if(found)
		printf("Found");
	else
		printf("Not found");
}
```

b)

```
int MEMBER(int* A, int S, int LOW, int HIGH){
	int i;
	int found=0;
	for(i=LOW;i<=HIGH;i++)
		if(A[i]==S)
			return(1);
	return(0);
}
```

c)

```
#include<stdio.h>
int MEMBER(int* A, int S, int LOW, int HIGH){
	int i;
	int found=0;
	for(i=LOW;i<=HIGH;i++)
		if(A[i]==S)
			return(1);
	return(0);
}
int main(){
	int AVAILABLE[100]; /* not interested in where the contents come from */
	int SELECTED[10];

	int i;
	int different=1;
	for(i=0;i<10;i++)
		if(MEMBER(SELECTED,SELECTED[i],i+1,9))
			different=0;
	int appear=1;
	for(i=0;i<100;i++)
		if(!MEMBER(AVAILABLE,SELECTED[i],0,99)) /* if element is not anywhere AVAILABLE … */
			appear=0; /* the elements of SELECTED do not all appear in AVAILABLE */
	if(different && appear)
		printf("Correct"); /* actual message not specified */
	else
		printf("Error");
}
```

Examiners' Notes

Although this was not a popular answer in section A many of the candidates that attempted the question produced a correct or near correct solution for part a). In part b) a typical error was not to use the parameters 'LOW' and 'HIGH' within the for loop.

In part (c) the majority of candidates that attempted this question produced poor or incomplete solutions, by not using the 'MEMBER' function created in part b) to check that all elements of the 'SELECTED' array were different and to check that all elements of the 'SELECTED' array also appeared in the 'AVAILABLE' array.

A2

{ please see exam paper for question }

Answer pointers

```
int THROW[5]; /* holds the number on the top face of each of the 5 dice */
int COUNT[7]; /* COUNT[0] not used. If n of the dice show the number i then COUNT[i]=n */
int INCOUNT(int c){ /* does the count c appear in the array? */
	int i;
	for(i=1;i<=6;i++){
		if(COUNT[i]==c)
			return(1);
	}
	return(0);
}
int SUMTHROW(){ /* total of the numbers on the top face of all 5 dice */
	int sum=0;
	int i;
	for(i=0;i<=4;i++){
		sum+=THROW[i];
	}
	return(sum);
}
int main(){
	int i;
	for(i=1;i<=6;i++){ /* initialise COUNT */
		COUNT[i]=0;
	}
	for(i=0;i<=4;i++){ /* compute COUNT */
		COUNT[THROW[i]]++;
	}
	int score;
	if(INCOUNT(5)) /* 5 of a kind */
		score=50;
	else if(INCOUNT(4)) /* 4 of a kind */
		score=30;
	else if(INCOUNT(3)&&INCOUNT(2)) /* 3 of a kind and 2 of a kind */
		score=25;
	else
		score=SUMTHROW(); /* sum of the 5 faces */
}
```

Examiners' Notes

Not many candidates attempted this question and less than 20% of these candidates achieved a pass mark.

In many cases the candidate copied out the question which gained them no marks or entered the code to determine the scores for 5 of a kind, 4 of a kind, or 3 of a kind and 2 of a kind without adding any details as to how the number shown on the top of the dice was counted. Few candidates calculated the sum of the 5 dice thrown.

A3

{ please see exam paper for question }

Answer pointers

a) Trace

		0	1	2	3	4	5		G	H	J	G<V[J]	V[J]<H	||
	V	7	6	5	4	3	2		3	5				
if											0	true	false	true
then		Y												
if											1	true	false	true
then			Y											
if											2	true	false	true
then				Y										
if											3	true	true	true
then					Y									
if											4	false	true	true
then						Y								
if											5	false	true	true
then							Y							

b) change "or" into "and" (|| into &&)

c)

```
char S[10]; /* the String, array of characters */
void classify(){
        int i;
        for(i=0;i<10;i++){
                if('A'<=S[i] && S[i]<='Z') /* note <= and && */
                        S[i]='U';
                else if('a'<=S[i] && S[i]<='z')
                        S[i]='L';
                else if('0'<=S[i] && S[i]<='9')
                        S[i]='D';
                else
                        S[i]='O';
        }
}
```

Examiners' Notes

This was a popular question attempted by three-quarters of the candidates, many of which received a pass mark. Many candidates traced the function successfully; however typical errors were to assume that the symbol '||' indicated 'OR' rather than 'AND'.

In part c) several candidates misunderstood the question and solved the problem for the example array given in the question for which they lost marks and others found it difficult to determine whether characters were uppercase, lowercase or digits.

Valid alternative approaches to the part c) answer pointer shown above were rewarded, such as:

- *the use of standard functions (isupper(), islower() and isdigit())*
- *the testing of characters against a range of ASCII data for upper case, lower case and digits*
- *the use of switch statements against a list of upper case, lower case and digits to solve the problem.*

A4

{ please see exam paper for question }

Answer pointers

a)

```
int a(int b){
        int c,d;
        d=0;
        for(c=10;c>b;c--)
                d=d+1;
        return(d);
}
```

b)

i) identifiers have to start with a letter then continue with any number of letters and digits
[Note sometimes languages allow other characters like underscore]

ii) use "meaningful" identifiers – choose pronounceable words that mean something in the context of the program

c)

identifiers	a b c d
constants	0 10 1
conditional expression	c>b
iterative statement	for(c=10;c>b;c--) d=d+1;

the statement that is repeated d=d+1;

d)

```
int a(int b){
        int c,d;
        d=0;
        c=10;
        while(c>b){
```

```
            d=d+1;
            c--;
        }
        return(d);
}
```

Examiners' Notes

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

Part a) was well answered although many candidates lost marks by not indenting their code. In part b) marks were lost by many candidates who were unaware of all the rules that apply in choosing a legal identifier in a modern programming language; however, the majority showed a good understanding of how a programmer might choose a "good" identifier.

The evidence suggests that few candidates were able to provide a correct conditional expression in part c) and there was a lack of understanding regarding a "while loop" in part d), where many candidates confused their answer with a "for loop" using:

while(c=10; c>b; c- -){ rather than while (c>b){

SECTION B

Candidates were required to answer **FIVE** out of the eight questions set

B5.

{ please see exam paper for question }

Answer pointers

(Pseudocode or program code accepted)

```
PROGRAM Decimal2Binary
        Variable Declarations
        INTEGER DecimalNumber, Quotient, i, j
        INTEGER ARRAY BinaryNumber
        BEGIN
                OUTPUT ("Enter any positive decimal number: ")
                INPUT "DecimalNumber"

                Quotient ← DecimalNumber
                i ← 1

                WHILE (Quotient != 0) DO                    {Loop while Quotient not zero}
                        BinaryNumber[i] ← Quotient MOD 2    {Calculates Remainder}
                        Quotient ← Quotient / 2
                        i++
                ENDWHILE

                {LSB calculated first, so BinaryNumber needs to be printed out in reverse order}
                PRINT ("Equivalent binary value of decimal number:" DecimalNumber)

                FOR (J=i-1, j>0, j--)
                        PRINT BinaryNumber[j]
                ENDFOR
        END
```

Examiners' Notes

This was the least popular question in Section B and the answers provided were poor. Many candidates provided code that would only operate with the decimal number 19. Others ignored the requirement for input and output, including reversal of the remainder digits.

Other issues included logic errors and a failure to use white space, indentation and comments. Poor handwriting sometimes made it difficult/impossible for the examiners to identify commands and variables.

B6

{ please see exam paper for question }

Answer Pointers

a) A dry run test consists of the computer programmer examining each line of source code one step at a time to ensure that it performs as expected and to find any errors. It is a manual test that does not involve computers.

b) White box testing involves examining the structure of the code and running tests that exercise all pathways through the code. The programmer would prepare test data and predicted results ready for comparison with the actual results.

One of the main limitations of white box testing is that it is not practical to test each and every path of the loops in large systems. Furthermore, the source code needs to be available to plan the testing. The focus of white box testing is therefore on the internal logic and pathways within the program and does not link this logic to the requirements of the specification. Any missing functionality will not be revealed by white box testing.

c) The following skills are needed for white box testing:

- A detailed knowledge of the program logic of the source code being tested.
- The ability to determine the test cases needed to cover the various pathways through the program. This involves the identification of an input, an action or an event, and the expected response.
- The ability to interpret the test results and to identify the location of any errors.

Examiners' Notes

Many candidates provided meaningless and generalised comments for part a) and failed to mention that dry run testing is concerned with the manual desk checking of program code.

Part b) was answered far better, with many candidates able to demonstrate their understanding of white box testing. However, a significant number forgot to discuss the limitations.

Part c) was answered well by the better prepared candidates. Others either omitted this part altogether or repeated answer points used in Parts a) and b).

B7

{ please see exam paper for question }

Answer Pointers

a) Random Access means the process of transferring information or reading from a memory location in which every memory location can be accessed directly, rather than being accessed in a fixed sequence.

b) Computer main memory (RAM), hard disk backing storage or flash drives (alternative terminology accepted).

c) A typical scenario is where an individual, fast, online response is required from a database query; examples could include bank balance or mortgage queries. Scenarios related to batch processing were not appropriate.

Examiners' Notes

This was not a popular question and generally candidates did not gain high marks. Many wrote about RAM for all parts of their answer, even though the question was primarily concerned with random access rather than random access memory.

B8

{ please see exam paper for question }

Answer Pointers

a) Typical sorting methods likely are: Bubble Sort, Insertion Sort, Quick Sort.

b) The paragraph below is an outline description of a Bubble Sort:

Bubble sort is a simple sorting algorithm that works by repeatedly stepping through the list to be sorted, where it compares each pair of adjacent elements and swaps them if they are in the wrong order. This process is repeated until no swaps are needed, which indicates the list is fully sorted. Bubble sort can be used to sort arrays in either ascending or descending order.

The iterations needed can also be counted, so if there are (n) elements in the array, the process will need to be repeated a maximum of (n-1) times. This is the simplest sorting algorithm, but it is inefficient for large lists.

Note: Any valid sorting algorithm and appropriate description was credited.

Examiners' Guidance Notes

This question was generally well answered, with most candidates having a clear idea of the method. Although some candidates labelled their answer as "Bubble Sort", the actual description was of other types of sort or even search algorithms.

Marks were deducted for incomplete or over-generalised descriptions.

B9

{ please see exam paper for question }

Answer pointers

a) The binary system operates in base 2 and uses digits the 0 to 1, whereas the octal system operates in base 8 and uses the digits 0 to 7.
b) Two popular methods of searching are Binary Chop and Linear Search
c) Microsoft Windows 8 is an example of Systems software and Visual C++ IDE is an example of Computer Programming tools.
d) The two phase of the software development life cycle that follow design are Implementation and Testing
e) The acronym OOP stand for Object Oriented Programming
f) A compiler is a program that translates source code into machine code.
g) In a program comments are included for the human reader, to help them understand the code operation.
h) The basic control structures used in computer programs are sequencing, selection and repetition.

i) Debugging is an important part of programing and is used to detect the runtime errors that cannot be detected by the compiler.

j) Two popular methods of texting programs are black-box and white box.

k) An example of a procedural language is C and an example of an object-oriented language is C++

l) Two kinds of buttons that tend to occur in groups on web-based forms are radio buttons and check boxes.

Examiners' Notes

A popular question and many candidates gained high marks. Apart from e), all the sentences required two entries for completion. A mark was only given where both entries were correct.

Credit was given for valid alternative answers. For example, in part d) "coding" was accepted instead of "implementation". This is because some textbooks refer to the implementation stage taking place after coding and testing have been completed and signed-off.

B10.

{ please see exam paper for question }

Answer Pointers

a) Stack: a collection of data elements; all data elements are added to the top of the stack and deleted from the top of the stack / allows access to the last data element inserted only.
Queue: a collection of data elements where data elements are added to the "rear" of the queue and deleted from the "front" of the queue.
Difference: stacks are a Last in First out structure (LIFO), whereas queues are a First in First out structure (FIFO).

b) Sequential programming: program states the exact order in which processes are to be executed.
Parallel programming: multiple tasks executing at the same time.
Difference: sequential more common, easier to learn. Obvious communication between tasks, as new task inherits everything left by previous task. Communication between parallel tasks is difficult, as working out whether a group of parallel tasks have all finished can be problematic.

c) System software: designed to offer easy access to features provided by hardware (e.g. file access by name, provides all the hardware addressing and load, save).
Application software: designed to enable user to accomplish common user tasks (e.g. in word processing provides WYSIWYG editing).
Difference: system software interfaces directly with hardware, whilst application software interfaces with system software and users.

Examiners' Notes

Some candidates did not attempt Part a) or confused the two terms. Some candidates confused sequential and parallel processing with parallel running or RAD development where programs are developed in parallel. Most provided adequate answers for part c).

B11

{ please see exam paper for question }

Answer pointers

Part	Line	Error Explanation	Error type
a	7	"k" not declared	compile time

b	6	index not allowed in i[1] as i is not declared as an array	compile time
c	12	last time round loop; i is 9, I + 1 is 10, so v[i+1] is invalid	run time
d	8	")" missing	compile time
e	13	variable required (in place of 5)	compile time
f	10	Invalid type v[0] boolean expression required after if	compile time

Examiners' Notes

The correct line number, a relevant error explanation and the type of error (compile or run time) were needed to gain the maximum two marks available for each part.

One mark was given if the candidate provided the correct line number plus either: a correct error explanation or the correct error type.

No marks were given for line numbers alone or for error explanations and error types without line numbers.

The evidence suggests that many candidates lost marks by not reading the question correctly.

B12

{ please see exam paper for question }

Answer pointers

A sketched form could show:

i. a method for the new user to enter their name (TEXT BOX)
ii. a technique to enter the age of the user, for example: 0-15, 16-35, 36-55 (DROP DOWN MENU)
iii. a method to enter the gender of the user (RADIO BUTTONS)
iv. a method for the user to show a combination of methods by which they are willing to be contacted (for example: phone, email, letter) (CHECK BOXES)
v. a method to enter the completed registration form details onto the system (SUBMIT BUTTON)

Alternative answer points:

- Alternative to (i) might be TEXT AREA, but only need one line for a name
- Alternative to (ii) might be RADIO BUTTONS, but would take up a lot of space
- Alternative to (iii) might be DROP DOWN MENU but this is usually used when there are many options.
- Alternative to (iv) might be a multi-choice menu but not commonly used, as confusing and more likely to make mistakes
- No real alternative to (v) the SUBMIT BUTTON

Examiners' Notes

Two marks were given for each of the five elements and a further two marks for the overall design of the form. No credit was given for using the same element twice.

Most candidates gained high marks. Candidates lost marks for either not answering the questions as set or for using the same elements for each of the inputs, for example using mainly text boxes or radio buttons.