BCS HIGHER EDUCATION QUALIFICATIONS
Level 4 Certificate in IT

March 2012

EXAMINERS' REPORT

Software Development

General comments on candidates' performance

There were some excellent papers and those candidates should be congratulated. However, many appeared to have little knowledge and understanding of software development and the following advice is offered to help future candidates:

- ***Study past examiners' reports and practise writing code.*** *The best way to learn a programming language is to program. Use the questions set in past examinations to practise writing and executing small sections of code. Also, it would be worthwhile writing small programs to carry out simple everyday tasks, for example: a program to check the interest due on a savings account, or one to analyse personal expenditure.*

- ***Consider what the question requires.*** *Although there were some excellent answers that gained full marks, many candidates lost marks by providing answers, including copious lines of program code, that were unrelated to the question. Candidates are advised to read each question carefully and ensure they understand exactly what the examiners are seeking before pen is put to paper.*

- ***Write clearly and neatly.*** *Programming is a discipline that requires precision and care when writing code. Unfortunately, many candidates took little care, often crossing through coding and using arrows to insert afterthoughts. Examiners often had difficulty identifying which variables or commands were being used and candidates lost marks as a result.*

- ***Avoid repetition.*** *Some candidates attempted to answer questions by merely repeating the same points or the code used to answer a previous question part. Candidates should be advised that no marks will be gained for such efforts.*

- ***Answer all parts of the questions attempted.*** *Some candidates omitted parts of the question, even when they appeared to have the required knowledge.*

- ***Follow instructions.*** *The examination requires candidates to answer two questions from Section A and four from Section B. No credit is given for answering more than the required number of questions.*

Please note that, for future examinations, Pascal extracts will not be used. Answers to programming questions will be accepted in any reasonable language; however, C and languages based on its syntax are recommended.

SECTION A

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

A1.

a) Write a section of code to read an array *v* (starting at the first element) and which copies all the entries of value 5 or more from array *v* to consecutive locations in array *w*.

Example

Array *v*	3	9	5	1	2	6	4	8

Array *w*	9	5	6	8	0	0	0	0

(8 marks)

b) Convert the code you have written in part a) into a function (or procedure) called *closeUp* which has a parameter called *min* such that no elements of *v* below min are copied to *w*. The result of the function is the number of elements copied to *w*. (In the case of a procedure, return the number of elements copied to *w* as a second parameter.)

(6 marks)

c) Write a function to find the average of the values in array w from the index *low* to the index *high*

(8 marks)

d) Write a complete program using your answers to b) and c) which reads 100 values from a file into array *v*, calls *closeUp* with parameter 10 and then prints out the average of the non-zero values from array w

(8 marks)

Answer pointers

Trying to guide students into writing a complete program with 'subroutines'
Happy for arrays to be handled as globals.
CloseUp algorithm requires two indexes to be maintained being the current visiting point in v and the current loading point in w

a) (8 marks)

```
int vp;
int wp=0;
for(vp=0;vp<8;vp++)                 //go through v
        if(v[vp]>=5)                //check element value big enough
                w[wp++]=v[vp];      //copy to w
```

b) (6 marks)

```
int closeUp(int min){               //this time as a function with param
        int vp;
        int wp=0;
        for(vp=0;vp<8;vp++)
                if(v[vp]>=min)
                        w[wp++]=v[vp];
        return wp;                  //return the number of elements copied
}
```

c) (8 marks)

```
float average(int low,int high){
        int i;
        for(i=low;i<=high;i++)          //for the specified elements
                sum+=w[i];              //find total
        return(sum/(high-low+1.0));     //return average as a float
}
```

d) (8 marks)

```
#include <stdio.h>
int v[100], w[100];
void readv(int n){                      //read specified number of values into v
        int i;
        for(i=low;i<n;i++)
                scanf("%d",&v[i]);
}
void main(){
        int wp;
        readv(100);                     //populate array v
        wp=closeUp(10);                 //copy larger values to w
        ave=average(0,wp-1);            //compute average
        printf("%f",ave);               //and print
}
```

Examiners' Guidance Notes

Not many candidates attempted this question, even so there were some good solutions created that gained high marks by creating the complete program using subroutines closeup and average.

Many candidates produced a correct or near correct response for part (a) but then didn't manage to complete or submit an answer for part (b & c) subroutines or the complete program required in part (d).

A2. A University department requires a report generator for the progression of students from one year to the next. Write a program to produce this report in line with the following specification.

Students take 12 courses each. The actual data is available in a text file called *results*. Each line in the file corresponds to one student and lists their *student_id* (integer) and their 12 marks (each as a percentage, rounded to an integer). There are 50 students in the class.

The students need to achieve a minimum performance and to gain 100 credits to progress. Minimum performance means scoring 20% or more in every module. Ten credits are awarded for each course for which the student gains 40% or more.

In order to make your program most useful, you should arrange that the key figures in these rules (in this example, 50, 100, 20%, and 40%) are easily changeable in your program so that it can be used with other students for whom the rules are different.

In the resulting report each line should report one student. There should be 6 columns as shown below.

Student_ id	overall average	total credits	total credit>=100 Yes/No	Minimum performance Yes/No	Progress to next year Yes/No
101265	55.5	100	Yes	No	No
231498					
333444					

(30 marks)

Answer pointers

Most marks for answers with solutions using subroutines parameterised by mimimumCredit (100), minPerfThreshold (20), creditPoint (40), classSize (50)

```
#include <stdio.h>;
const int CLASS_SIZE=50;
const int MIN_PERF_THRESHOLD=20;
const int CREDIT_POINT=40;
const int CREDIT_PER_MODULE=10;
FILE * fp;
void process_student(){
        int id;
        char min_perf='Y';
        int total_mark=0;
        int credits=0;
        int mark;
        int m;
        fscanf(fp,"%d",&id);
        for(m=0;m<12;m++){                      //loop through 12 courses for student
                fscanf(fp,"%d",&mark);
                total_mark=total_mark+mark;             //Sum total marks
                if(mark<MIN_PERF_THRESHOLD)min_perf='N';
                if(mark>CREDIT_POINT)credits+=CREDIT_PER_MODULE;
        }
        printf("%d ",id);                       // print student id
        printf("%4.1f ",total_mark/12.0);       // print average
```

```
        printf("%3d ",credits);                                 // print total credits
        if(credits>=100)printf("%c ",'Y'); else printf("%c ",'N');
        printf("%c ",min_perf);                                 // print other logic columns
        if(credits>=100 && min_perf=='Y')printf("%c ",'Y');
        else printf("%c ",'N');
        printf("\n");
}
void progression_list(class_size){
        int i;
        fp=fopen("results","r");
        for(i=0;i<class_size;i++)
                process_student();
}
void main(){
        progression_list(CLASS_SIZE);
}
```

Examiners' Guidance Notes

Few candidates attempted this question and generally they created poor quality answers.

A few candidates created workable solutions making use of constant parameters and subroutines to simplify the code, using an approach similar to the answer pointer given above. Other candidates attempted a solution for one student rather than the class size of 50

Many candidates simply wasted their time by writing out the question and in some cases they created flow charts as an overall solution to describe the logic needed.

A3. Consider the program segment below and then answer the questions that follow.

Version in C	Version in Pascal

void f(int a){
  int b, c; char d;
 	
  b = 0; c = a - 1;
  while(b < c){
  d = v[b];
  v[b] = v[c];
  v[c] = d;
  b++;
  c--;
  }
}
	procedure f(a : integer);
	var b, c : integer; d : char;
	begin
	  b := 0; c := a - 1;
	  while b < c do begin
	  d := v[b];
	  v[b] := v[c];
	  v[c] := d;
	  b := b + 1;
	  c := c - 1
	  end
	end;

a) Trace the execution of the call f(8) when the array *v* is initialised as follows. The first element of the array *v* is at index 0.

Array *v*	'Q'	'W'	'E'	'R'	'T'	'Y'	'U'	'I'

Show the final state of the array v.

(16 marks)

b) State in your own words the overall effect of the function f

(4 marks)

c) Rewrite f making any changes that you think would make the code better for a human reader

(10 marks)

Answer pointers

f(8) means a=8
b=0 c=7
0<7 true — QWERTYUI
swap v[0],v[7]
b=1 c=6
1<6 true — IWERTYUQ
swap v[1],v[6]
b=2 c=5
2<5 true — IUERTYWQ
swap v[2],v[5]
b=3 c=4
3<4 true — IUYRTEWQ
swap v[3],v[4]
b=4 c=3
4<3 false — IUYTREWQ

v starts as QWERTYUI
v ends as IUYTREWQ

b) Overall effect is to reverse the elements of v

c) Require well-chosen identifiers and some comments

```
void reverse(int len){ //reverse the elements of array v, length len
        int low=0, high=len-1; char swap;
        while(low<high){
                swap=v[low];
                v[low]=v[high];
                v[high]=swap;
                low++;
                high--;
        }
}
```

Examiners' Guidance Notes

This was a popular question with some candidates achieving high marks.

a) Some candidates did not trace the subroutine call, and lost marks by simply providing part or the whole of the "right answer". In other cases candidates traced through systematically, but omitted to show the final state of the array v.

b) Many candidates correctly described the reversal of the elements in the array; whilst others described the effect of each line of code without giving any indication of the overall effect of the subroutine.

c) In a few cases candidates chose new meaningful names for the function and the variables which helped to make the code more understandable; whilst in other cases the original code was simply copied out with some identifiers modified that added nothing to improving the clarity of the code. In-program comments were rarely used and then they tended to be superficial rather than explaining features like the swapping of array elements.

A4. Consider the following program and answer the questions that follow.

Version in C	Version in Pascal

char a; float b;
char c(int d, int e){
  float f;
 	
  f = d / e;
  if(f < 1){
  b = b + f;
  return('y');
  }else
  return('n');
}
int main(){
  b = 0;
  a = 'y';
  printf("%c", g);
  printf("%c", c(2, 3));
}
	program p;
	var a : char; b : real;
	function c(d : integer; e :
	integer):char;
	var f : real;
	begin
	  f := d / e;
	  if f < 1 then begin
	  b := b + f;
	  c := 'y';
	  end else
	  c := 'n';
	end;
	begin
	  b := 0;
	  a := 'y';
	  write(a);
	  write(c(2, 3))
	end.

In the program above find and copy out ALL the (different)

a) integer constants
b) string or character constants
c) local variables
d) global variables
e) formal parameters
f) relational operators
g) type identifiers
h) actual parameters
i) arithmetic operators

(2 marks x 9)

Notes: In parts (a-i) only copy out what is asked for - do not include any of the surrounding context. If an item occurs more than once in the program then it is only necessary to write it down once.

From the program above, find and copy out ONE example of

j) an assignment statement
k) a compound statement
l) a conditional statement
m) a function call

(3 marks x 4)

Answer pointers

i)

a) 0,1,2,3
b) 'y', 'n'
c) f,g
d) a,b
e) d,e
f) <
g) char, float, int

h) 2,3
i) /,+

ii)

j) f=d/e
k) {b=b+f;return('y');}
l) if(f<1){b=b+f;return('y');}else return('n');
m) c(2,3)

Examiners' Guidance Notes

This was a popular question that was well answered by the majority of candidates.

In some cases candidates copied out whole sections of code in an attempt to gain the marks for the longer questions (j, k, l, m).

Surprisingly few candidates were able to provide a compound statement or a complete conditional statement.

SECTION B

(Candidates were required to answer five out of the eight questions set)

B5. a) In the programming language of your choice, state how the operations of integer division (division with integer result) and modulo (remainder after division) are carried out? State the language used.

(2 marks)

b) Write a subroutine called *binary* (in the language used in part a) to take an integer parameter and output its value in binary.

For example the call *binary*(29) outputs 11101

(10 marks)

Answer pointers

a) In C,
integer division uses operator '/' with integer operands
modulo uses operator '%'

b)

```
#include <stdio.h>
void binary(int d){
        char b[10]; /*save binary digits here for reversal later*/
        int i=0;
        while(d>0){
                rem=d % 2; /*modulo*/
                if(b[i++]==1)b[i++]='1'; else b[i++]='0';
                d=d / 2; /*integer division*/
        }
        i--;
        while(i>=0){ /*backwards through array*/
                printf("%c",b[i]);
                i--;
        }
}
```

Examiners' Guidance Notes

This was not a popular question and there were some poor attempts. Some candidates provided code that would only operate for the number 29. Other candidates, who appeared to understand about divisions and remainders, inexplicably failed to answer part a) of the question.

The examiners were not concerned about the detail of the input and output used, as long as the relevant verbs were used. Any fixed size for the array used was acceptable, provided it was above five.

Many candidates failed to reverse the binary digits at the end. The highest marks were given to candidates who reversed the binary digits in the array, used relevant comments and utilised appropriate indentations.

B6. By writing two functions called factorial and factorial_r show how the calculation known as factorial may be computed by a non-recursive method and a recursive method.

(12 marks)

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

Answer pointers

```
int factorial(int n){
        if(n==0) return 1;
        else return n*factorial(n-1);
}

int factorial_r(int n){
        int res=1;
        int i;
        for(i=1;i<=n;i++)
                res*=i;
        return res;
}
```

Examiners' Guidance Notes

Although there were some sound answers, a number of candidates merely wrote down the factorial of various numbers and failed to provide any code. Others omitted the requirement to provide both recursive and non-recursive answers.

Many candidates did not attemt this question.

B7. Carry out the following tasks in **either** C **or** Pascal

a) Write code using a while loop which is equivalent to the following *for* loop.

Version in C	Version in Pascal

for(i=0;i<=99;i++)
  v[i]=0;
	for i:=0 step 1 to 99 do
	  v[i]:=0;

b) Rewrite the following conditional code without using the logical operators &&(and), ||(or), !(not)

Version in C	Version in Pascal

if(p && !q)
  x = 0;
else
  x = 1;
	if p and not q then
	  x := 0
	else
	  x := 1

c) Write code to find the maximum value in array *v* leaving the answer in variable *max*.

Version in C	Version in Pascal
int v[100]; int max;	var v:array[0..99] of integer, max:integer;

(3 x 4 marks)

Answer pointers

a)

```
i=0;
while(i<=99){
        v[i]=0;
        i++;
}
```

b)

```
if(p)
        if(q) x=1;
        else x=0;                   // p && !q --> 0
else
        x=1;
```

c)

```
int i, max, v[100];
...
max=v[0];for(i=1;i<100;i++)if(v[i]>max)max=v[i];
```

Examiners' Guidance Notes

This was a popular question, with most candidates gaining 3-4 marks for parts a) and b). Part c) was less well answered, with many failing to understand what was required.

B8. Answer the questions by referring to **either** the C **or** the Pascal version of the code below:

Version in C	Version in Pascal
int f(int j,int k){int i,m;m=0;for(i=0;i<=7;i++)if(j<=v[i]&&v[i]<=k)m++;return m;}	function f(j,k:integer): integer;var i,m:integer; begin m:=0;for i from 0 to 7 do if j<=v[i] and v[i]<=k then m:=m+1;f:=m end

a) Comment on the presentation/layout of this code?

(2 marks)

b) Rewrite the code to make it look more conventional.

(4 marks)

c) Could the original 'bad' code run successfully without the change you made in b)? Why?

(2 marks)

d) Explain in your own words what the code is trying to achieve.

(2 marks)

e) Apart from what you did in b), suggest one more improvement to the code.

(2 marks)

Answer pointers

a) Programmer has not used white space to layout the code tidily/the layout is unconventional.

b)

```
int f(int j,int k){
        int i,m;
        for(i=0;i<8;i++)
                if(j<=v[i]&&v[i]<=k)m++;
        return m;
}
```

c) Yes it would run successfully - the compiler does not take any notice of white space

d) Find the number of values in v that lie between j & k

e) i) Add comments
ii) Give more meaningful names to identifiers
iii) Combine the two for loops into one

Examiners' Guidance Notes

This was another popular question, with most candidates gaining some marks. Parts a) and e) were often left unanswered, even though the candidate appeared to have a good understanding of the code and the issues involved. Candidates lost marks in part c) by copying the code inaccurately or for failing to include indentations. Part d) was not always answered clearly, although the examiners gave the benefit of doubt if there was some understanding of a count being maintained. Credit was given for alternative answers for part e).

B9. a) What do you understand by the term 'debugging'?

(4 marks)

b) In a simple programming environment where the programmer has only the standard output facilities of the programming language to use, how is debugging approached?

(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. Debugging includes the identification and correction of errors.

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

Candidates generally understood the concept of debugging and usually gained 3-4 marks for part a). However, parts b) and c) were poorly answered, with candidates often repeating the points made in part a) or stressing the need for care to be taken when coding, or extolling the virtues of using firewalls and anti-virus software. A disappointing response to a straightforward question.

B10. Supply the words to complete the following sentences:

a) To translate and run a program a ____ or an ____ is needed.
b) The first 2 phases of the Software Development Life Cycle are ____ & ____.
c) The acronym WIMP stands for ____, ____, ____, ____.
d) The list of common programming language paradigms includes functional languages, ____ & _____.
e) Depending on the hardware device involved, a computer program may have ____ access or ____ access to a file.
f) A computer program is translated from ____ language to ____ language.
g) The two kinds of buttons that tend to occur in groups on web forms are called ____ & ____.
h) The acronyms LIFO and FIFO stand for ____ & ____.
i) The basic control structures found in computer programs are sequencing, ____ & ____.
j) The basic (or primitive) data types in a programming language include integer, ____ & ____.
k) Apart from basic data types there are constructed types which include ____ & ____.
l) Two popular methods of testing programs are ____ & ____.

(1 mark x12)

Answer pointers

a) To translate and run a program a **compiler** or a **interpreter** is needed
b) The first 2 phases of the Software Development Life Cycle are **requirements** and **design**
c) The acronym WIMP stands for **windows**, **icons**, **menus**, **pointer**
d) The list of common programming language paradigms includes, **procedural languages** & **object-oriented languages**
e) Depending on the hardware device involved, a computer program may have **sequential** access or **direct** access to a file
f) A computer program is translated from **high-level** language to **low-level** language
g) The two kinds of buttons that tend to occur in groups on web forms are called **radio buttons** & **check boxes**
h) The acronyms LIFO and FIFO stand for **Last-In-First-Out** & **First-In-Last-Out**
i) The basic control structures found in computer programs are sequencing, **selection** & **repetition**
j) The basic (or primitive) data types in a programming language include integer, **float** & **character**
k) Apart from basic data types there are constructed types which include **arrays** & **records** / **structs**
l) Two popular methods of testing programs are **black-box** & **white-box**

Examiners' Guidance Notes

This was another popular and straightforward question, with most candidates gaining adequate marks. Where relevant, alternative answers were accepted, although there was a surprising lack of knowledge – even for the candidates who had gained high marks in other questions.

B11. a) Name two algorithms that can be used for sorting an array.

(2 marks)

b) Choose ONE of the algorithms named in a) and describe how it works.

(10 marks)

(Note: code is not required)

Answer pointers

a) Expect Bubblesort, Insertion Sort, Quick Sort, etc

b) Students will do well to choose a simple algorithm to describe - eg Bubblesort

i) Go through the array looking at adjacent pairs and swapping if out of order
(this is the bubble and sweeps the largest element to the end of the array)

ii) Repeat this process a sufficient number of times to ensure the array is fully sorted
(this can be counted (ie if the array has length n, repeat n-1 times in total
or
until a bubble process happens without a swap

Examiners' Guidance Notes

Candidates either knew appropriate sorting algorithms and gained high or full marks, or had little knowledge and gained nil or low marks. Although the question stated that code was not required, many candidates provided code alongside an explanation. Others provided an explanation together with arrays of values to illustrate the changes at each stage of the iteration.

Weak candidates confined their answers to the principles of algorithms and failed either to identify or describe a sorting algorithm.

B12. a) What is the difference in programming terms between an array and a record?
(3 marks)

b) Give typical examples of
i) data that would be handled as an array
ii) data that would be handled as a record
iii) data that would be handled as a combination of array and record
(3 x 3 marks)

Answer pointers

a)
An array is a collection of information **all of the same type**
Access to the elements is by a **numerical index - a[i]**

A record is a collection of information **of possibly different types**
Access to the elements is by a **named field - r.f**

b)
i) the marks for each student in the class (array of marks)
ii) the contact info for a student (record with fields: name, address, email, phone)
iii) the contact details for a class of students (array of records)

Examiners' Guidance Notes

Candidates displayed a surprising lack of knowledge for this question and, for those who appeared to have the required knowledge and understanding, the explanations and examples used were weak or non-definitive.

Frequently, part a) was answered only in the context of an array, although hardly any candidates explained how arrays were indexed. The requested examples for the three sections of part b) generally lacked detail, or the examples given were insufficient for the examiner to assess whether the candidate actually understood the differences.