THE BCS PROFESSIONAL EXAMINATION
Professional Graduate Diploma

September 2012

EXAMINERS' REPORT

SOFTWARE DEVELOPMENT

General comments on candidates' performance

There were some excellent papers and those candidates should be congratulated. However, a significant number of candidates had little knowledge and understanding of software development.

Future candidates are advised to study this subject using past examiners' reports, and to practise writing and executing small sections of code. This could be simple programming tasks, such as calculating compound interest on a savings account, or the more complex activity of coding common mathematic algorithms.

Although there were some excellent answers that gained full marks, many candidates lost marks by providing answers 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 they answer the question.

Many candidates lost marks by using careless syntax. They must understand that programming is a discipline that requires precision and care when writing code. Examiners often had difficulty identifying which variables or commands were being used and 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.

A1.
{please include question}

Indicative answer pointers

Trying to guide students into writing a complete program with 'subroutines'
Happy for arrays to be handled as globals
The marking algorithm is straightforward conditional arithmetic

a)

```
float diff;
int i;
for(i=0;i<8;i++){
        diff = abs(experiment[i]-correct[i]) * 100 / correct[i];
if( diff<1 ) score[i]=8;
else   if( diff<10 ) score[i]=4;
else score[i]=0;
}
```

b)

```
void marking(high,tight,low,loose){
float diff;
int i;
for(i=0;i<8;i++){
                diff = abs(experiment[i]-correct[i]) * 100 / correct[i];
if( diff<tight ) score[i]=high;
else   if( diff<loose ) score[i]=low;
else score[i]=0;
}
}
```

c)

```
float total(count,high){
        int sum=0;
for(i=0;i<count;i++)
                sum+=score[i];
        return (sum*100)/(high*count);
}
```

d)

```
#include <stdio.h>
void main(){
        FILE * fp;
        fp=fopen("correct","r");
for(i=0;i<8;i++) scanf("%d",&correct[i]);
        fp=fopen("experiment","r");
for(i=0;i<8;i++) scanf("%d",&experiment[i]);
marking(8,1,4,10);
printf("%f", total(8,8);
}
```

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 functions or subroutines.

Many candidates produced a correct or near correct response for parts (a & b), typical errors being to not find the absolute value of variable diff.

Few candidates managed to complete or submit a correct answer for either part (c) or the complete program required in part (d).

A2.

{please include question}

Indicative answer pointers

More marks were allocated for students who broke the problem down into subroutines

Read the year1 file line by line, calculating student average and storing in year1ave array
Ditto year2

Compare corresponding elements in across year1ave and year2ave to find the largest increase from year1 to year2

```
#include <stdio.h>
void main(){
        FILE * y1p, y2p;
        int sum1, sum2;
        int improved=0;
        int improvement=0;
int i, m;
int mark;
float step;
y1p=fopen("year1","r");
        y2p=fopen("year2","r");
        for(i=0;i<50;i++){
                sum1=0;
sum2=0;
for(m=0;m<12;i++){
        fscanf(y1p,"%d",&mark);
sum1+=mark;
        fscanf(y2p,"%d",&mark);
sum2+=mark;
                }
                step=sum2/12.0-sum1/12.0;
                if(step>improvement){
improvement=step;
improved=i;
                }
        }
        printf("%d",improved);
}
```

Examiners' Guidance Notes

This was not a popular question but there were some high marks gained by the few candidates who made a good attempt at reading data from the text files and looping through 50 students and 12 sets of results.

The majority of candidates who attempted this question ignored the fact that the question stated the data was held in two text files. Instead they wrote out pages of code to prompt and to input the data from the keyboard. Other candidates attempted a solution for one student rather than the class size of 50

A3

{please include question}

Indicative answer pointers

a) bubble(8) means n=8

				0	1	2	3	4	5	6	7
i=1	v[0]>v[1]	9>7	true								
array v				7	7	5	3	2	4	6	8
i=2	v[1]>v[2]	7>5	true								
array v				7	5	5	3	2	4	6	8
i=3	v[2]>v[3]	5>3	true								
array v				7	5	3	3	2	4	6	8
i=4	v[3]>v[4]	3>2	true								
array v				7	5	3	2	2	4	6	8
i=5	v[4]>v[5]	2>4	false								
array v (unchanged)											
i=6	v[5]>v[6]	4>6	false								
i=7	v[6]>v[7]	6>8	false								

b) The 2 assignments after the "if" should perform a swap - but they don't
Declare temporary variable e.g. swap (add "int swap;" into the code), change the two assignments to

```
swap=v[i-1];
v[i-1]=v[i];
v[i]=swap;
```

c) Once the bubble function is fixed it will bubble the largest element of the array to the end. The simplest way to achieve a full sort is to repeat the bubble, n times

```
void sort(int n){
    int i;
    for(i=0;i<n;i++)bubble(n);
}
```

There are various ways to make this more efficient - e.g.
i) escaping from the sort loop when no swaps took place in the last bubble
ii) decrease the span of the bubble by one element on each repeat

Examiners' Guidance Notes

This was a very popular question with many candidates achieving high marks in part a) where they successfully traced the sort as 7 5 3 2 2 4 6 8. Some candidates tabulated the trace which made it easy to follow, whilst others copied out and evaluated the code for each run through the loop Typical errors were caused by incorrect evaluation of the IF statement.

Most candidates who attempted part b) discovered the mistake and added a holding statement so that the swap was successful.

Some candidates successfully found a solution for part c) by repeating the sort 8 times using either a For Loop or a While Loop.

A4.

{please include question}

Indicative answer pointers

i)

a)	d,e
b)	b,main
c)	+, *
d)	0,100
e)	int, void
f)	, ;
g)	if, for
h)	< <=

ii)

i)	c<0
j)	for(d=0;d<=c;d++)e=e+d*d;
k)	if(c<0)c=-c;
l)	b(100)

Examiners' Guidance Notes

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

In part i) many candidates showed an understanding of the program, but lost marks by not including each item for a particular question, for example they added local variable "d", but did not include local variable "e".

Part i) included eight options rather than the intended nine. The marks for this section were scaled to reflect the missing option.

In some cases candidates copied out whole sections of code in an attempt to gain the marks for the longer questions in part (ii). Surprisingly few candidates were able to provide a function call or complete conditional statement and iterative statements.

B5.

{please include question}

Indicative answer pointers

a)

```
int     binary[10];
int     decimal(int len){
        int d=0;
        for(i=0;i<len;i++){
                d=d*2+binary[i];
        }
        return d;
}
```

b)

```
void main(){
        int count=0, b;
        printf("Input digits"); // 1 or 0, -1 to finish
        b=scanf("%d",binary[i]);
        while(b>=0){
                binary[count]=b;
                count++;
                scanf("%d",binary[i]);
        }
        printf("Converted to decimal gives %d", decimal(count));
}
```

Examiners' Guidance Notes

Although this represented an interesting exercise in binary logic and programming, few candidates attempted the question. Those who did were confused by the fact that the MSB (most significant bit) was the first element of the array when usually the first element is the LSB (least significant bit).

B6

{please include question}

Answer pointers

```
int fibonacci_r(int n){
        int i;
        if(n==0)return 1;
        if(n==1)return 1;
        return fibonacci_r(n-1)+fibonacci_r(n-2);
}

int fibonacci(int n){
        int i;
        int fn2;
        int fn1=1;
        int fn=1;
        if(n==0)return 1;
        if(n==1)return 1;
        for(i=2;i<=n;i++){
                fn2=fn1;
                fn1=fn;
                fn=fn-1+fn-2;
        }
        return fn;
}
```

Examiners' Guidance Notes

Again, few candidates attempted this question. Generally those who did misunderstood the mathematics of the Fibonacci sequence and this was reflected in the code produced.

B7

{please include question}

Indicative answer pointers

a)

```
for(i=100;i>0;i--)f(i);
```

b)

```
if(p)   x=0;
else    if(q) x=1;
        else x=0;
```

c)

```
char save=v[0];
for(i=0;i<4;i++)v[i]=v[i+1];
v[4]=save;
```

Examiners' Guidance Notes

Part a) of the question was quite a simple exercise: the conversion of a while loop to a for loop. Candidates who knew the required syntax found this quite easy.

Although part b) required the candidate to rewrite the expression to avoid using !, || and &&, many candidates ignored this requirement and lost marks as a result.

Although not too complex, very few candidates attempted part c) which required a simple loop to cycle the array elements back by one position. A number of candidates lost the first element of the array by inadvertently overwriting it and therefore failed to gain the maximum marks available.

B8

{please include question}

Indicative answer pointers

a) Programmer has not used white space to lay out the code tidily.

b)
```
int v[8];
int f(){
        int i,j,k;
        j=v[0];
        for(i=0;i<8;i++)
                if(v[i]<j)j=v[i];
        k=v[7];
        for(i=7;i>=0;i--)
                if(v[i]>k)k=v[i];
        return k-j
}
```

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

d) Find the spread of v (the difference between the max & the min values).

e) Improvements could include:

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

Examiners' Guidance Notes

This was a popular question which was generally answered well, with several candidates gaining full marks. Most gained some marks for part b) which required candidates to lay out the code in a conventional manner. However, many lost marks by combining statements on one line or by not copying the code accurately.

Part d) was not answered well, as most candidates were unable to understand the function of the code.

B9

{please include question}

Indicative 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

Another popular question, which was generally answered well, although some answers were too brief to attract the highest marks or the candidates had confused unit and integration testing with other testing regimes, such as regression testing or the use of test harnesses.

B10.

{please include question}

Indicative answer pointers

a) Two popular methods of searching are Linear Search & Binary Chop
b) The binary system operates in base 2 whereas the hexadecimal system operates in base 16
c) The acronyms ROM and RAM stand for Read-Only Memory & Random Access Memory
d) Computer memory is usually measured in units of GB(GigaBytes) while processor speed is measured in MHz (MegaHertz)
e) In a program the parts introducing new identifiers are called Declarations and the parts only intended to be read by the human reader are called Comments
f) Run time errors can be found in code by suing a Debugger which installs Breakpoints at selected points
g) Two diagrammatic notations used in the design of programs are Flowcharts & UML Diagrams
h) Whitespace is any selection of program text made up of Space, Tab & Newline
i) The alternative names for the data structures called LIFO and FIFO are Stack & Queue
j) The last two phases of the software development life cycle are Testing & Maintenance
k) Two ways to obtain a one-from-many selection from the user on a web form are Drop-Down Menu (ListBox) & Radio Buttons
l) An example of a procedural language is C and an example of an object-oriented language is C++

Examiners' Guidance Notes

A popular question that was not well answered, even though the examiners accepted valid alternative answers to those shown in the answer pointers above.

B11.

{please include question}

Indicative answer pointers

a) Expect maybe Linear Search and Binary-Chop

b) Linear Search (looking for needle)
    No special requirements of array
    Start at one end and compare each element with needle

```
        Stop when match found or end of array met

    Binary Chop (looking for needle)
        Array must first be sorted (eg into ascending order)
        i)
            a) if no elements remaining - finish - not found
            b) Choose the middle element of array and compare with needle
        ii)
            a) If match - needle found - finish
            b) If needle is smaller, reject upper half of array and repeat from i) with
            remaining elements
            c) If needle is larger, reject lower half of array and repeat from i) with remaining
            elements
```

Examiners' Guidance Notes

Although the question referred to a search algorithm, a significant number of candidates provided answers concerning sort algorithms and gained no marks for their efforts. Candidates are advised to read each question carefully and make sure they understand the requirements.

Generally this question was answered well, with many candidates gaining the maximum marks.

B12.

{please include question}

Indicative answer pointers

a) A record is used where a collection of data of dissimilar types is required to be handled as a single unit

b) [Rest of answer given for two different languages Javascript and C]

Declaring and using a single record in Javascript

i)
```
var rec={}
rec.fld1='a';
rec.fld2=1;
rec.fld3=2.3e-4;
```

ii)
```
v = rec.fld1;
```

c) Declare an array of records
Then index the array to select one record, then choose a field of the record

```
var rec_array=[ { fld1:'a',fld2:1,fld3:2.3e4 }, { fld1:'b',fld2:2,fld3:3.4e5 } ]
alert( rec_array[0].fld1 );
```

In C

b)

i)
```
typedef struct {char fld1;int fld2;float fld3;} rectype;
rectype rec;
void main(){
        rec.fld1='a';
```

```
                rec.fld2=1;
                rec.fld3=2.3e-4;
            }
```

ii) `v = rec.fld1`

c)

```
typedef struct {char fld1;int fld2;float fld3;} rectype;
rectype rec_array[100];
void main(){
        rec_array[0].fld1='a';
        rec_array[0].fld2=1;
        rec_array[0].fld3=2.3e-4;
}
```

Examiners' Guidance Notes

A significant number of candidates had little idea of the concept of a record, with a few believing it was a written record (i.e. documentation of a program).

Although part b) clearly required the candidate to provide a simple example in a programming language, many candidates provided continuous prose rather than the requested code.

Part c) was not attempted by many and there was some evidence of candidates running out of time.