THE BCS PROFESSIONAL EXAMINATION

Professional Graduate Diploma

September 2014

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 number of candidates lost marks needlessly 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. Also, many candidates answered more than the required number of questions. Future candidates should note that no credit is given for answering additional questions.

Although for this sitting some answers were acceptable in pseudocode, this did not mean that the need for care and precision, necessary for programming, could be ignored. All too 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

Answer pointers

The code below has been run and tested

Cannot have associative arrays in C so have to store distances in array indexed by integers and use another array of town names to provide a key.

The function strcmp (from the string library) compares 2 strings and produces a 3 way result (-1, 0, +1) depending on whether 1st string is earlier, same or later alphabetically than 2nd.

The function strcpy copies a string from 1st param to 2nd.

If a student did not use these functions but used similar functions with 'sensible' names then that was acceptable. The code is simpler if town names are 'simplified' to 1 letter each - so less marks.

```
#include "stdio.h";
#include "string.h";
//triangular table is here stored as the above diagonal elements of a square array
float DISTANCES[5][5]={
	{0.0,  10.0,  8.0,   20.0,  15.0},
	{0.0,  0.0,   5.0,   30.0,  12.0},
	{0.0,  0.0,   0.0,   40.0,  9.0},
	{0.0,  0.0,   0.0,   0.0,   50.0},
	{0.0,  0.0,   0.0,   0.0,   0.0}
};
//the key linking a town name to an index is given by the TOWNS array
char* TOWNS[5]={"Axminster","Bakewell","Canterbury","Doncaster","Exmouth"};

//A route is stored in the ROUTE array (town names have a max length of 20)
char ROUTE[5][20];

int lookupTown(char* town){ //convert town name to index - this is simpler if only
use single letter
	int i;
	for(i=0;i<=5;i++)
		if(strcmp(TOWNS[i],town)==0) //0 means the 2 strings match
			return i;
	return -1;
}
float lookupDistance(int from, int to){
	int temp;
	if(from>to){ //only have a triangle of data, need to>from
		temp=from;
		from=to;
		to=temp;
	}
	return DISTANCES[from][to];
}
int isTown(char* town){
	return strcmp(town,".")!=0;
}
void readRoute(){ //read a succession of town names, finishing with string which is
just "."
	int i=0;
	int len;
	char town[20];
	printf("Type town names as route...\n");
	do{
		scanf("%s",town);
		len=strlen(town);
		printf("town %s, len %d\n",town,len);
		strcpy(ROUTE[i],town);
		i++;
	}while(isTown(town));
}
float routeLength(){ //calculate length of route given ROUTE as an array of town
names
	int from, to;
	int i=0;
	float distance=0.0;
	if(!isTown(ROUTE[i]))
		return distance;
	from=lookupTown(ROUTE[i]);
	while(isTown(ROUTE[i+1])){
		to=lookupTown(ROUTE[i+1]);
		distance+=lookupDistance(from,to);
		i++;
		from=to;
	}
	return distance;
}
int main(){
	float distance;
	int i;
	readRoute(); //read into array ROUTE
	while(isTown(ROUTE[0])){ //keep finding length of routes while user types
them in
```

```
            i=0;
            printf("ROUTE\n");
            while(isTown(ROUTE[i])){
                  printf("%s,",ROUTE[i]);
                  i++;
            }
            distance=routeLength();
            printf("%f",distance);
            readRoute();
      }
}
```

Examiners' Guidance Notes

This was one of the least popular questions in Section A and was attempted by approximately 12% of candidates. In a few cases the candidate produced a near correct answer creating a structured solution based on functions and town names handled as strings.

However the quality of answers was generally poor and marks were mainly gained for determining some of the distances between towns using a range of conditional statements on towns A to E.

Many candidates simply copied out the question which was time consuming and gained them no marks.

A2

Answer pointers

This code has been run and tested. The question does not ask for a main program

```
#include "stdio.h";
//Note that the first entry in each array initialisation is a dummy to fill
subscript 0
float upToWeight[]={0,5.0,10.0,25.0,50.0,100.0};
float postCost[]={0,1.50,2.00,3.00,4.00,5.00};
long int
ISBN[]={0,9123450870214,1111111111111,2222222222222,3333333333333};
float bookWeight[]={0,12.49,7.00,4.00,11.00};
long int ORDER[]={0,9123450870214,2222222222222};
float lookupWeight(long int bn){
      int i=1;
      while(ISBN[i]!=bn)i++;
      return bookWeight[i];
}
float findCost(float w){
      int i=1;
      while(upToWeight[i]<w)i++;
      return postCost[i];
}
float orderCost(){
      int i;
      float totalCost=0;
      for(i=1;i<=10;i++)
            totalCost+=findCost(lookupWeight(ORDER[i]));
      return totalCost;
}
int main(){
      printf("Total Cost: %f\n", orderCost() );
}
```

Examiners' Guidance Notes

This was the least popular question in Section A, but there were some high marks gained by the few candidates who made a good attempt at all three parts of the question. The vast majority did not consider declaring or initializing data for some elements of the arrays for up-to-weight, post code, etc.

Many candidates attempted this question by describing their approach but not actually writing any code. As in question (1), many candidates simply copied out the question which was time consuming and gained them no marks.

A3

Answer pointers

a) Trace the call of the function *f*. (10 marks)

c	d	a (c)	a (d)	a(c) == a (d)	b
0	1	a(0)=3	a(1)=7	3==7	0
0	2	a(0)=3	a(2)=5	3==5	0
0	3	a(0)=3	a(3)=1	3==1	0
0	4	a(0)=3	a(4)=2	3==2	0
0	5	a(0)=3	a(5)=7	3==7	0
1	2	a(1)=7	a(2)=5	7==5	0
1	3	a(1)=7	a(3)=1	7==1	0
1	4	a(1)=7	a(4)=2	7==2	0
1	5	a(1)=7	a(5)=7	7==7	1
2	3	a(2)=5	a(3)=1	5==1	1
2	4	a(2)=5	a(4)=2	5==2	1
2	5	a(2)=5	a(5)=7	5==7	1
3	4	a(3)=1	a(4)=2	1==2	1
3	5	a(3)=1	a(5)=7	1==7	1
4	5	a(4)=2	a(5)=7	2==7	1

b) State what you observe to be the effect of *f*. (6 marks)

Effect: counts the number of duplicate entries in array

Only one duplicate in array

c) Give the function *f* a new name which is more appropriate. (2 marks)

New name: countDuplicates

d) Using the renamed function write a function *isSet* which returns 1 if the values in the array form a set (i.e. there are no repeated values) and 0 otherwise. (6 marks)

```
function isSet – see program below
```

e) Write a more efficient version of *isSet*, not using the original *f* or its renamed version, which can return the 1 or 0 answer more quickly than the *isSet* of (d) (6 marks)

```
more efficient isSet – see isSet2 below
```

```
#include "stdio.h"; //to use printf
int a[6]={3,7,5,1,2,7}; //initialise array a
int f(){
     int b=0,c,d;
     for(c=0;c<=5;c++)
          for(d=c+1;d<=5;d++){
               if(a[c]==a[d])
                    b++;
               printf("c=%d,d=%d,b=%d\n",c,d,b); //simulate
trace
          }
     return b;
}
int main(){
     f();
     printf("isSet:%d\n",isSet()); //version 1
     printf("isSet2:%d\n",isSet2()); //version 2
}

int countDuplicates(){return f();} //rename f to
countDuplicates

int isSet(){ //version using count
     if(countDuplicates()>0)return 0; else return 1;
}

int isSet2(){ //faster version, not using f
     int i,j;
     for(i=0;i<=5;i++)
          for(j=i+1;j<=5;j++)
               if(a[i]==a[j])
                    return 0; //by short-circuiting the loops
we get the 0 answer faster
     return 1;
}
```

Examiners' Guidance Notes

This was a very popular question with many candidates achieving some marks in part a) where they attempted to trace through the code provided. In a few cases all stages of the trace were completed and the b=1 was returned. 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. Many candidates did not continue to trace though the code beyond the point b=1 was found.

In parts b) and c), many candidates identified that the effect of the code was to find the duplicates in the array whilst others thought that it was some kind of search.

Few candidates attempted part d). In most cases they nearly created a satisfactory solution, although they mixed up the return 0 and return1 statements. Few candidates were able to produce a workable solution of part e), in most cases they copied or made minor modifications to the "matching" code from part a) of the question.

A4

Answer pointers

a)

```
int f(char*v){
        int i=0;
        while(i<10){
                if(v[i]=='.')
                        return i;
                i++;
        }
        return -1;
}
```

b) White space is space, tab, newline characters, inserted by the programmer to make programs more readable

c) A sequence of statements appears inside {} . It is traditional to put each statement on a separate line and indent them all to the same depth. If a statement sequence appears inside another statement sequence then the inner one is indented further. This way the programmer's eye is guided when reading the program to be able the see the extent of any statement sequence.

d)

identifiers(3): f,v,i
constants(4): 0,10,'.',-1
operators(4): =,<,==,++
Boolean expressions(2): i<10, v[i]=='.'
A loop statement: while(i<10){if(v[i]=='.')return i;i++;}

e)

```
int f(char*v){
      for(i=0;i<10;i++)
      if(v[i]=='.')
      return i;
      return -1;
}
```

Examiners' Guidance Notes

This was a very popular question that was attempted by 93% of the candidates; it was generally well answered with approximately 60% of the candidates achieving a satisfactory pass mark.

Most of the candidates were able to gain maximum or near maximum marks for part a) and generally all showed a reasonable understanding of part b).

Marks were mainly lost in part d) where many candidates found it difficult to distinguish between Boolean expressions and loop statements.

Surprisingly few candidates were able to provide a satisfactory replacement for the while-loop code with a for-loop in part e).

SECTION B

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

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

B5.

Answer Pointers

Pseudocode Example (accept any valid alternative or programming solution)

```
Pseudocode Convert Binary to Decimal
Variable Declarations

CHARACTER ARRAY Binary[16]          /* Binary Number Input

INTEGER N                           /* Number of Binary Digits Input

INTEGER Decimal                     /* Decimal Number Output

BEGIN

        Decimal ← 0                 /* Initialise Decimal Total at zero

        PRINT "Enter number of binary digits – Max 16 digits"
        INPUT (N)

        PRINT "Enter binary digits Most Significant Bit first"

        FOR (i=N TO 0 Step -1)
                INPUT Binary[i]     /* Enter Binary Digit into Array

                Decimal ← Decimal + (Binary[i] x 2^[i])

                IF (i > 0) PRINT "Enter Next Binary Digit"
        NEXT
                PRINT "Decimal Conversion = "Decimal
END
```

Examiners' Guidance 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 "1011" example provided in the question. Others ignored the requirement for input and output. There were many handwriting issues, which made it difficult for the examiners to identify commands and variables.

B6

Answer Pointers

a) An insertion sort is an iterative process, where one element is removed from the input data, the location the element belongs in is then determined and the element is inserted into that location; smaller elements are successively shifted to

the left until they are larger than the element on their left at which point they are inserted. This process is repeated until no elements of input data remain.

b) Insertion sort advantages include: simple implementation, efficient for small data sets, more efficient than bubble sort as less scans involved and very efficient where the data in a list is substantially sorted.

Disadvantages include: insertion sort needs a large number of element shifts which is inefficient for large lists, as the number of elements is increased the performance of the program will decrease.

c) Insertion Sort Pseudocode

```
INSERTION SORT
Variable Declarations
Array A[N]                              /* Array with N elements for sorting
Integer i, j                            /* loop index variables

    FOR (j=2 TO N) DO
        Key ← A[j]                      /*Insert element A[j] into the sorted sequence
        i ← j -1

        WHILE (i > 0 AND A[i] > Key) DO
            A[i+1] ← A[i]               /* Swap elements in array
            i ← i - 1
        END
        A[i+1] ← key
    END
```

Examiners' Guidance Notes

Only a small number of candidates attempted this question and the marks gained were low. Many only attempted parts a) and b), and confused the insertion sort with other sorting algorithms, such as the bubble sort. This confusion continued into part c), with many providing code for alternative sorting algorithms or code that was incapable of execution.

B7

Answer Pointers

a)

i) Linux OS	- Systems Software
ii) Internet Explorer Web Browser	- Application Software
iii) Windows Visual C++ IDE	- Computer Programming Tools
iv) Dreamweaver Web Design Software	- Application Software
v) Java Interpreter	- Computer Programming Tools
vi) Mac OS X	- Systems Software

b)

Systems software - the collection of programs that make up the operating system which is concerned with shielding the user from the details of the hardware

Computer programming tools - the software, such as compilers and linkers that are used to translate and combine computer source code and libraries into executable programs.

Applications software - the computer software that causes a computer to perform useful tasks beyond the running of the computer itself, typically it is pre-packaged solutions to perform a specific task, for example a spreadsheet or word processor.

Examiners' Guidance Notes

This was the most popular question in Section B and most candidates gained high marks. Part a) was straightforward and answered well, although some candidates wasted time by explaining the function of each software item. Candidates also gained high marks for part b), although no marks were given for simplistic comments such as "programming tools are used for programming" or "systems software runs the systems".

B8

Answer Pointers

a) Sequential programming uses one processor and all the tasks are performed in sequence one after another.

Parallel programming uses multiple processors so that several tasks can be carried out simultaneously [usually with some coordination]

b) Searching problems - different tasks search different parts of the search space simultaneously; numerical problems e.g. matrix multiplication - one task per element of the answer matrix

Ideally need a problem which breaks down into several (many) identical sub-problems which are independent

Examiners' Guidance Notes

The differences were misunderstood by most candidates. Many confused the terms with project management or system implementation, often referring to project teams working on different aspects of a project at the same time or mentioning parallel running. Only the best candidates suggested problems, such as weather forecasting, where parallel programming could be usefully employed.

B9

Answer pointers

a)

name of student	- text box	- textual input, not known in advance
gender	- radio button	- 1 out of n choices (where n is small)
subjects	- check boxes	- multiple choice of n
course	- menu	- 1 out of n choices (where n is large)

b)

A simple sketch showing what the form elements look like on a browser screen, with some labelling for clarification was acceptable

Examiners' Guidance Notes

Most candidates provided acceptable answers. Some lost marks by failing to provide a reason for the form element used, or by omitting one or more of the form elements. A wide variety of sketches was produced but many candidates ignored the requirement for labelling.

B10.

Answer Pointers

a) Compile-time error: syntax error, grammatical error in program
Run-time error: computational error, divide by zero, infinite loop
Difference: compile time errors always come before run-time errors

b) Sequential access file: pointer in file, only forward progress allowed, get to desired point via every record on the way. If gone too far then only option is to rewind and start again from beginning.
Direct access file: can go directly to any record in file irrespective of where the last visit was made
Difference: sequential access simpler but slower

c) High-level language (HLL): programmer insulated from properties of machine (store size, word size, actual memory addresses). Can define named variables and data structures and use sophisticated control structures
Low-level language (LLL): m/c language or assembly language. Have to deal with actual m/c instructions and registers and store addresses.
Difference: HLL suits humans, LLL suits computer. Can get HLL auto translated to LLL

Examiners' Guidance Notes

Another popular question, although many candidates omitted to explain the difference between the terms. Most demonstrated some understanding of compile-time and run-time errors, although some of the explanations were too simplistic, for example, compile-time errors occur at compile time. There was confusion with the different types of access, with some referring to access control systems and security of data. Many completely misunderstood the difference between high level and low level languages, and some omitted this part from their answers.

B11.

Answer Pointers

Part	Line	Error Explanation	When Found
a	5	Identifier “w” not declared	compile time
b	10	index not allowed in x[0], x is not declared as an array	compile time
c	13	1st time through loop z[x-1] is z[-1]; -1 is bad subscript	run time
d	9	")" missing	compile time
e	7	'w'= cannot assign to a constant	compile time
f	13	z[x-1]= cannot receive float (needs char)	compile time

Examiners’ Guidance Notes

Only the better candidates gained high marks for this question. There were many careless answers, where candidates had clearly not read or understood the requirements of the question.

For each part of this question, candidates needed to provide:

- The line number where the error would occur
- An explanation of the error
- Where the error would be identified (compile-time or run-time)

All of the above three were needed to gain two marks. Some candidates omitted one of the three requirements; in this instance, one mark was given if the line number was provided plus one of the remaining two bullet points.

Alternative answers were accepted by the examiners if the candidate referred to earlier lines of code that could have been changed to prevent the error occurring.

B12

Answer Pointers

a) In the "Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially; the phases include: requirements analysis, design, implementation, testing and maintenance.

   The advantages of using the waterfall method are:

   - Easy to manage due to the rigidity of the model
   - Each phase has specific deliverables and a review process.
   - Phases are processed and completed one at a time.
   - Works well for smaller projects where requirements are very well understood.
   - Clearly defined stages and well understood milestones.

b) Software prototyping operates in a similar way to the traditional waterfall method except that the developer repeatedly loops through the Requirements, System Design and Coding until a satisfactory solution is obtained.

The advantages of using software prototyping are:

- Ideal for use where the users are unable to accurately specify their information system requirements
- Improves communication between system developers and client
- Helps to identify confusing functions or even missed functionality
- Reduces risk of failure, as possible risks are identified quickly and precautions can be taken to solve the problem.
- Ideal approach for developing software by non-computing specialists

Examiners' Guidance Notes

There were many good answers, although many candidates ignored the requirement for advantages, or wasted time writing about disadvantages. Prototyping software appeared to be better understood than the waterfall method.