BCS THE CHARTERED INSTITUTE FOR IT
BCS HIGHER EDUCATION QUALIFICATIONS
BCS Level 4 Certificate in IT

September 2013

EXAMINERS' REPORT

Software Development

General comments on candidates' performance

The standard of candidates' answers for this examination was higher than for the March sitting, 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 time reading and understanding the question before writing their answers.

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

A1

Indicative answer pointers

```
#include<stdio.h>
#define STUDENTNO 100
#define COURSENO 12
int rank[STUDENTNO];
float averageRank[COURSENO];
float agreement[COURSENO];
void readCourse(){
        int i;
        for(i=0;i<STUDENTNO;i++){
                scanf("%d",&rank[i]);
        }
}
int sq(int k){return(k*k);}
void analyse(int c){
        int i, sum=0;
        for(i=0;i<STUDENTNO;i++){ /* compute average rank first */
                sum+=rank[i];
        }
        averageRank[c]=sum/STUDENTNO;
        agreement[c]=0; /* can't start on agreement until average known */
        for(i=0;i<STUDENTNO;i++){
                agreement[c]+=sq(rank[i]-averageRank[c]);
        }
}
```

```
int min(float* v){
	int i;
	int smallest=v[0];
	for(i=1;i<COURSENO;i++)
		if(v[i]<smallest)
			smallest=v[i];
	return(smallest);
}
void main(){
	int c;
	for(c=0;c<COURSENO;c++){
		readCourse();
		analyse(c);
		printf("Average ranking for course %d is %f", c, averageRank[c] );
	}
	printf("Course with lowest average ranking is %d", min(averageRank) );
	printf("Course with the most agreement is %d", min(agreement) );
}
```

Examiners' Guidance Notes

This was the least popular question in Section A. In a minority of cases the candidate produced a correct or nearly correct answer making use of functions and clearly defined variables to allow the number of students and courses to be easily modified if required.

However the quality of answers was generally poor and marks were mainly gained for determining the average ranking of the courses; surprisingly few candidates attempted to calculate the number of the course with the most agreement even though the "mean square difference" used to determine this agreement was described in the question.

A2

Indicative answer pointers

[Note: This answer is written out in C which does not handle strings very well. This question is intended to be a test of information handling, not the finer points of a particular programming language.

So the following points were accepted:

- If the answer uses the type string as though it were built-in
- If the student writes if(s==v[i]) to test if 2 strings are equal instead of strcmp(...)

```
#include<stdio.h>
#include<string.h> /* this is needed for strcmp */
typedef char* string;
string Fruit[100], different[100];
int popularity[100], count=0;

int occurs(string s,string *v,int m){
	int i;
	for(i=0;i<m;i++)
		if(strcmp(s,v[i])==0) /* 0 means the 2 strings are the same */
			return(i);
	return(-1);
}
void findDifferent(){
	int i,pos;
	for(i=0;i<100;i++){
```

```
                pos=occurs(Fruit[i],different,count);
                if(pos<0){ /* not found, this is a new fruit */
                        different[count]=Fruit[i];
                        popularity[count]=0;
                        count++;
                }else{ /* found before so just adjust popularity */
                        popularity[pos]++;
                }
        }
}
int max(int* v){
        int i,largest=v[0];
        for(i=1;i<count;i++)
                if(v[i]>largest)
                        largest=v[i];
        return(largest);
}
void main(){
        int i;
        /* Fruit=["apple","pear","banana","apple"]; */
        findDifferent();
        printf("The list of fruits mentioned");
        for(i=0;i<count;i++){
                printf("%s\n",different[i]);
        }
        printf("The most popular fruit is %d",max(popularity) );
}
```

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 Fruit array and looping through the 100 student choices to find the different fruits chosen and then the most popular choice; in a few cases they also were able to add satisfactory code to determine if there was a tie in popularity.

However, the majority of candidates who attempted this question ignored the fact that the question stated that the data was held in an array named Fruit. Instead they wrote out pages of code to prompt and to input the data from the keyboard. In some cases they then simply output a list of all the fruit in the array and not just the different types.

Some candidates misunderstood the question and attempted to provide a solution using the example data for array elements 0, 1, 2, 3, 4, and 98, 99 included in the question, rather than reading or using all of the elements of the Fruit array.

A3

Indicative answer pointers

a) Tracing the call of f('R','T','Y')

```
v1='R'
v2='T'
v3='Y'
i
0       'R'=='Q' && 'T'=='W' && 'Y'=='E'       No
1       'R'=='W' && 'T'=='E' && 'Y'=='R'       No
2       'R'=='E' && 'T'=='R' && 'Y'=='T'       No
3       'R'=='R' && 'T'=='T' && 'Y'=='Y'       Yes
return (3)
```

b) The purpose of this function was to search array A for a sequence of 3 elements that match v1,v2,v3 and return the index of the first of the 3 elements if found, or return -1 (not found) otherwise

c) The mistake in the function not shown up in the trace was that indexing exceeds the bounds of the array. This first happens when i=8 and the program accesses A[i+2]

d) The function can be corrected by

```
for(i=0;i<8;i++){ /* adjust the stop value of the for loop */
```

e) The advanced version of the function is written where the three parameters v1, v2 and v3 are replaced by two parameters V and N. In this version V is an array and the first N values of V are to be used like v1, v2 & v3.

```
int match(int* V, int N, int i){
        int k;
        for(k=0;k<N;k++)
                if(V[k]!=A[i+k])
                        return(0); /* false, no match */
        return(1); /* true, match found */
}
int f(int* V, int N){
        int i;
        for(i=0;i<8;i++){
                if(match(V,N,i)) /* does the whole of V match A from index i? */
                        return i;
        }
        return -1;
}
```

Examiners' Guidance Notes

This was a very popular question with many candidates achieving high marks in part (a) where they successfully traced the search and returned i= 3. 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 continued to trace though the code beyond the point where ('R','T','Y') had been found.

Many candidates who attempted part (b) thought that the code was a sort function rather than a search function.

In parts (c) and (d) some candidates continued the trace up to (i < 10) and found that the index exceeded the bounds of the array; the correction was then straightforward.

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

Indicative answer pointers

a) Human readable form below.

```
int fF(int k0){
        int i;
        if(k0>0)
                for(i=2;i<k0;i++)
                        k0=k0*i;
        return(k0);
}
```

b)
Defn i) Identifiers are made up of any mixture of lowercase letters and digits
Defn ii) Identifiers are made up of any mixture of letters optionally followed by a digit

"Any mixture of lowercase letters and digits" - is wrong because firstly upper case letters are allowed and secondly the identifier has to start with a letter

"Any mixture of letters optionally followed by a digit" - is wrong because it suggests only one digit is allowed - after the first letter, any mixture of letters and digits are allowed

c) Referring to the code in part (a):

i) all the different identifiers
- fF
- k0
- i

ii) all the different constants
- 0
- 2

iii) a conditional (logical, boolean) expression
- k0>0

iv) a conditional statement

```
if(k0>0)
        for(i=2;i<k0;i++)
                k0=k0*i;
```

v) the statement that is repeated by the loop

```
k0=k0*i;
```

d) Replacing the for-loop with a while loop.

```
i=2;
while(i<k0){
        k0=k0*i;
        i++;
}
```

Examiners' Guidance Notes

This was a popular question that was attempted by over 90% of the candidates, it was generally well answered and the vast majority of the candidates achieved a pass mark for this question.

Marks were mainly lost in part (b) where candidates seemed unaware that identifiers could contain a mixture of uppercase and lowercase letters and that a digit (or digits) could be included after the first letter.

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

SECTION B

B5

Indicative answer pointers

(a) Pseudocode Example (accept alternatives in program code)

```
Variable Declarations
INPUT ← Orders
        CASE Orders DIV 1000
                0: Discount ← 0.0
                1, 2: Discount ← 0.05
                3, 4: Discount ← 0.10
                5, 6, 7, 8, 9: Discount ← 0.20
              OTHERWISE
                Discount ← 0.25
        END CASE

        PRINT "Discount ="Discount *100 "%"
        Final_Cost ← Order * (1-Discount);
        PRINT "Final Cost = £ "Final_Cost;
END
```

(b) Pseudocode Example (accept alternatives in program code)

```
Variable Declarations
INPUT ← Orders
        IF (Order <= 999) THEN
                Discount ← 0.0
        ELSE IF (Order <= 2999) THEN
                Discount ← 0.05
        ELSE IF (Order <= 4999) THEN
                Discount ← 0.10
        ELSE IF (Order <=9999) THEN
                Discount ← 0.20
        ELSE
                Discount ← 0.25
        END IF
        PRINT "Discount ="Discount *100 "%"
        Final_Cost ← Order * (1-Discount);
        PRINT "Final Cost = £ "Final_Cost;
END
```

(c) The CASE statement is easier to read and to debug and the code runs more quickly than using multiple IF-THEN-ELSE statements.

Examiners' Guidance Notes

This was the least popular question in Section B. In general, the quality of answers was poor; many failed to answer Part a) of the question, confining their answers to Parts b) and c). A significant proportion of the candidates either omitted the requirement to output the final cost of the order or provided limited answers based on the example order value of £2,500.

B6

Indicative answer pointers

a) A bubble sort is a sort where adjacent items in the array or list are scanned repeatedly, swapping as necessary, until one full scan performs no swaps.

b) The main disadvantage of the bubble sort is that it can take a maximum of (N-1)scans to fully sort, where N is the size of the list that needs to be sorted; this is because an out of position item is only moved (or swapped) one position per scan.

c) Algorithm / Pseudocode

```
Variable Declarations
Array S with N elements for sorting (zero based)
Swapped ← True
j ←  0
WHILE (Swapped ==True) DO
        Swapped ← False
        j ← j + 1
        FOR (i=1 TO N-j) DO
                IF (S[i-1] > S[i]) THEN
                        Temp ← S[i-1]
                        S[i-1] ← S[i]
                        S[i] ← Temp
                        Swapped ← True
                END IF
        END FOR
END WHILE LOOP
```

Examiners' Guidance Notes

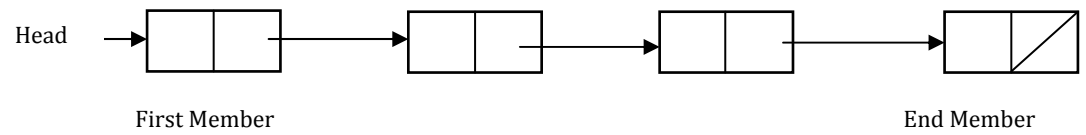
Around half of the candidates attempted this question, although many confined their answers to Parts a) and b). Part a) was answered well, with most candidates understanding in outline the operation of a bubble sort. Part b) was less well answered; most candidates realised that a bubble sort was inefficient but only the better candidates understood why.

Common errors in Part c) were failing to use a temporary store and logic errors. However, a few candidates provided excellent answers and gained full marks.

B7

Indicative answer pointers

(a) i) Linked List diagram



ii) Linked list data structure using C++

```
Struct ListNode
{
        Int data;
        ListNode *link;
};
```

iii) This structure is useful for storing a sequence of related data items where the number of these is not known initially and which changes dynamically as the run progresses.

(b) i) One dimensional array

Contents	10	24	8	30
Subscript	[0]	[1]	[2]	[3]

ii) int SmallArray[4]; // array of 4 integers named SmallArray and declared in C++

iii) This data structure is particularly useful in SORTING operations when all of these are in memory at the outset. The data in two locations can be swapped using the subscripts thus:

IF (item [1] IS GREATER THAN item [2]) THEN **swap** (Item[1],item[2]);
The values of the two subscripts [1, 2] can be varied systematically and data between any pair of locations in the array can be swapped for sorting purposes.

Examiners' Guidance Notes

Candidates who attempted this question were able to demonstrate their understanding and generally gained reasonable marks. However, some confined their answers to Part (a) i). Candidates frequently lost the second mark for Part (a) iii) and Part (b) iii), as they failed to demonstrate why a linked list or an array would be suitable.

B8

Indicative answer pointers

a) **Prototype** - a basic version of the system is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed. Prototypes can be of assistance where the user requirements are vague.

b) **4GL** - is a high level programming language or programming environment designed with a specific purpose in mind, such as the development of commercial business software

c) **Parallel processing** - the simultaneous processing of different tasks by two or more microprocessors, as by a single computer with more than one central processing unit.

d) A **data type** in programming languages is an attribute of a data which tells the computer (and the programmer) something about the kind of data it is. This involves setting constraints on the datum, such as what values it can take and what operations may be performed upon it.

Examiners' Guidance Notes

Although this question was attempted by almost 90% of candidates, the number reaching pass standard was lower than expected. Reasons included:

- *A limited knowledge of prototypes; many candidates provided examples unrelated to software development.*
- *Fundamental misunderstandings of parallel processing; many candidates confused this with parallel running.*
- *Limited knowledge of data types; some candidates provided non-programming examples.*

B9

Indicative answer pointers

3 marks available for **overall window design** layout (structure, frames, buttons, explanation). The labelled drawing should include all the interface elements used.

3 marks for each **interface element** (identifying element, drawing element, general or specific use).

Elements could include:

- Drop-down menu / list box/ combo box
- Radio buttons
- Check boxes
- Text box
- Command / Navigation buttons

Accept other interface elements.

Interface element explanations for general or specific use should be in the context of an online book store

Examiners' Guidance Notes

Another popular question and candidates usually achieved reasonable marks, although few gained the maximum 12 marks available. Reasons included:

- *Insufficient interface elements. Although the question specifically requested three interface elements, many answers gave one, two, or none at all.*
- *Inadequate or incomplete explanations. Many designs lacked any form of explanation.*
- *Diagrams not used. The question asked for a diagram to illustrate the layout.*

B10

Indicative answer pointers

a) Both compile time and run time are concerned with programming. **Compile time** is translating a high-level language program to low-level, whilst **run time** is the execution of the low-level instructions.

b) Systems software and applications software are different types of software. **Systems software** is designed to operate the computer hardware and provide a platform for running application software. A**pplication software** consists of programs designed to directly deal with solving the user's problems, such as word processing and accounting.

c) Both specification and design are stages of the SDLC. The **specification** states the user requirements and the **design** is about how these requirements will be achieved.

d) Queues and stacks are both ways in which data can be stored and processed. A **queue** is a store with two identifiable ends where elements join at one end and leave at the other, such that the first to join is the first to leave (FIFO). A **stack** is a store with one identifiable end where elements join and leave, such that the last to join is the first to leave (LIFO)

Examiners' Guidance Notes

This was the most popular question and most of whom had sufficient knowledge to achieve reasonable marks.

B11

Indicative answer pointers

a) **syntax error** occurs at compile time - something wrong in the phrase structure
run-time error - some unusual numerical situation has occurred; e.g. divide by zero

b) i) (2 / * 3) - expression - syntax requires only one (binary) operator between operands
ii) if(x>1 {y=1;} - statement - closing parenthesis missing after x>1
iii) a / (b - b) - expression - division by zero.

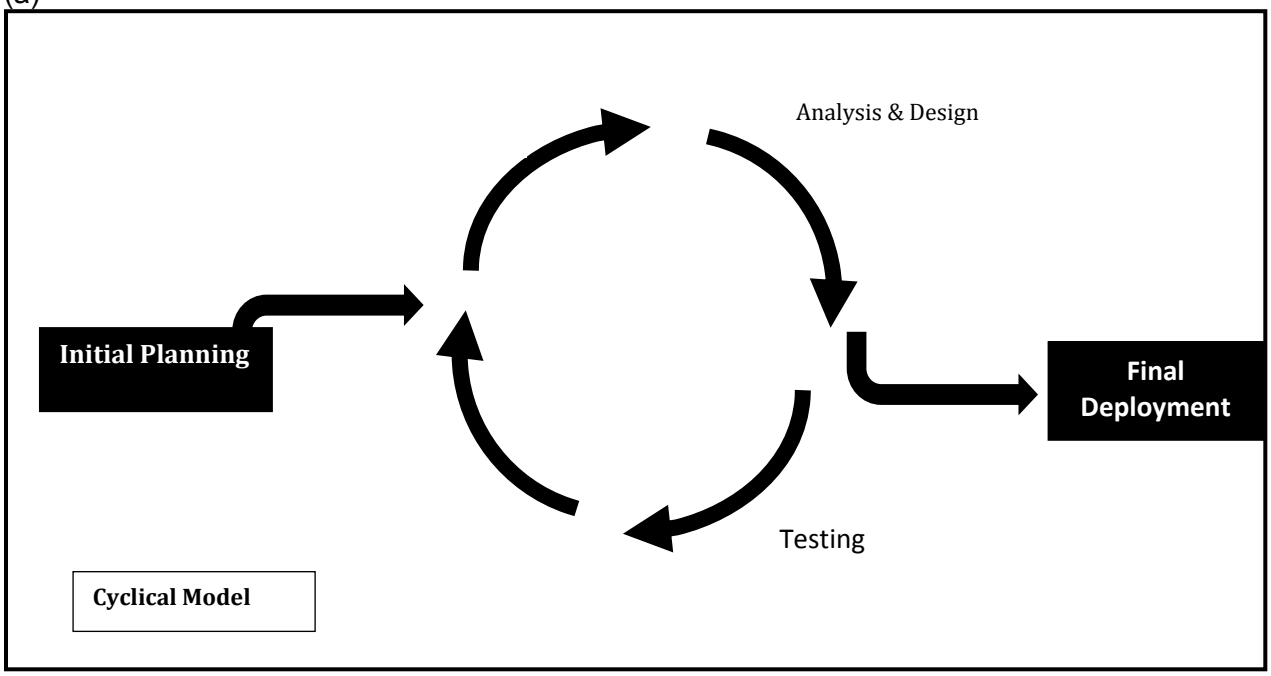
Examiners' Guidance Notes

Part a) was answered well but candidates lost marks for Part b) by either not answering this part at all or by failing to understand fully the requirements of the question. Two actions were required: the first was to provide lines of program code containing different types of error and the second was to describe the errors. Often candidates confined their answers to lines of code, without describing errors. Others provided descriptions that were too vague or which relied on other code that had not been supplied.

B12

Indicative answer pointers

(a)



(b) Brief explanation should include the following points:
The cyclical model is based on an incremental and iterative approach. Basically the requirements and the solution develop through collaboration between the client and the development team. It begins with an initial planning activity and then cycles through the phases from requirements to evaluation, until the customer is satisfied with the result, at which point the system is deployed.

(c) ONE advantage and ONE disadvantage from the following:

Cyclical Model Advantages	Cyclical Model Disadvantages
Decreased development time for project	Customer must have the ability to define the needs of the user.
Collaboration with customer ensure there are no misunderstandings	Documentation could be incomplete as it is normally carried out at later stages of project development.
Final end product is developed in the least possible time to meet customer needs	The software developers need to be highly skilled and good communicators to interact with the customer as well as to develop the software.

Valid alternative advantages & disadvantages accepted

Examiners' Guidance Notes

A number of candidates provided answers based around non-iterative development methodologies and gained no marks. However, most candidates knew and understood iterative development methodologies and gained high marks for all question parts.