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

March 2014

EXAMINERS' REPORT

Software Development

General comments on candidates' performance

The standard for this examination was higher than for recent sittings, indicating that candidates are taking notice of comments made in previous examination reports. However, failure to read the questions carefully and poor handwriting remain serious concerns; a significant number of candidates could have gained higher marks if they had addressed these issues.

Candidates are advised to spend time reading and understanding the exact requirements of the question before writing their answers. There were many incomplete answers or answers not relevant to the question set; these gained few, if any, marks. To give just one example, for question B11 candidates were required to state if errors in the supplied code would have been identified at compile time or run-time. Often this requirement was completely ignored, even though the candidate clearly had the required programming knowledge. The marks lost by omitting this simple part of the question could have made the difference between a pass and a fail.

For questions involving code or lists of instructions, many candidates lost marks through careless mistakes or indecipherable handwriting and crossings out, making it impossible for the examiners to see which variables and commands were being used or to understand the program structure. Programming is a discipline that requires precision and care when writing code.

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

Note that the question does NOT ask for any printing – just the program that does the calculations. Print statements are included here for completeness.

```
#include "stdio.h";
float TIMES[4][7]={
       {0,    0,0,0,0,0,0},
       {0,    22,33,11,55,66,44},
       {0,    2,3,1,5,6,4},
       {0,    9,8,9,8,9,8}
```

```
};
float min(float v[]){
	float min=v[1];
	int i;
	for(i=2;i<=6;i++){
		if(v[i]<min)
			min=v[i];
		}
	return min;
}
float max(float v[]){
	float max=v[1];
	int i;
	for(i=2;i<=6;i++){
		if(v[i]> max)
			max =v[i];
		}
	return max;
}
float minBM[4],maxBM[4];
float SCORES[4][7];
float TOTALS[7];
int main(){
	int i,r,c;
	float winningScore;

	printf("\n");
	printf("TIMES: \n");
	for(r=1;r<=3;r++){
		for(c=1;c<=6;c++)
			printf("%f\t",TIMES[r][c]);
		printf("\n");
	}
	printf("\n");

	for(i=1;i<=3;i++){
		minBM[i]=min(TIMES[i]);
		maxBM[i]=max(TIMES[i]);
	}

	printf("\n");
	printf("minBM:\n");
	for(r=1;r<=3;r++)
		printf("%f\n",minBM[r]);

	printf("\n");
	printf("maxBM:\n");
	for(r=1;r<=3;r++)
		printf("%f\n",maxBM[r]);

	printf("\n");
	for(r=1;r<=3;r++)
		for(c=1;c<=6;c++)
			SCORES[r][c]=(TIMES[r][c]-minBM[r])/(maxBM[r]-minBM[r]);

	printf("\n");
	printf("SCORES: \n");
	for(r=1;r<=3;r++){
		for(c=1;c<=6;c++)
			printf("%f\t",SCORES[r][c]);
		printf("\n");
	}
	printf("\n");

	for(c=1;c<=6;c++)
		TOTALS[c]=0;
	for(c=1;c<=6;c++)
		for(r=1;r<=3;r++)
			TOTALS[c]+=SCORES[r][c];

	printf("\n");
	printf("TOTALS: \n");
	for(c=1;c<=6;c++)
```

```
                printf("%f\t",TOTALS[c]);
        printf("\n");

        printf("\n");
        winningScore=min(TOTALS);
        printf("winning Score %f\n",winningScore);
        printf("\n");
}
```

Examiners' Guidance Notes

This wasn't a popular answer in section A although in a few cases the candidates that attempted the question produced a correct or near correct solution; in these cases typical errors included incorrect data type declarations.

However the majority of candidates that attempted this question produced poor quality or incomplete solutions by simply using conditional statements to calculate MAX and MIN and then calculating SCORES without making use of FOR loops.

In a few cases candidates simply wasted their time by writing out the question.

A2

Answer pointers

Note that 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
int UNSORTED[]={0,55,11,44,33,66,88,99,22,77}; //should really be up to index 100, but here
just 9
int SORTED[10];
int fullTo=0; //this records the first index in SORTED which is not filled
int findPosition(int v){
        int i;
        for(i=1;i<=fullTo;i++)
                if(SORTED[i]>v)return i;
        return fullTo;
}
void makeSpace(int p){ //shuffle all elements from p onwards 1 place to right
        int i;
        for(i=fullTo-1;i>=p;i--) //start at last filled index of array, work backwards
                SORTED[i+1]=SORTED[i];
}
void insert1(int v){
        int p=findPosition(v);
        makeSpace(p);
        SORTED[p]=v;
        fullTo++;
}
void insertionSort(){
        int i;
        for(i=0;i<10;i++)
                insert1(UNSORTED[i]);
}
int main(){
        int i;
        insertionSort();
        printf("SORTED\n");
        for(i=0;i<10;i++)
                printf("%d, ",SORTED[i]);
        printf("\n");
}
```

Examiners' Guidance Notes

Not many candidates attempted this question and only a quarter of these achieved the required standard. In many cases the candidate simply copied out the question which gained them no marks or attempted to sort the data in the UNSORTED array given in the question.

In a few cases candidates produced correct or near correct solutions for all the functions required; typical errors were made in the makeSpace function where it was necessary to make a space in the array by moving all of the elements from point p one place to the right.

A3

Answer pointers

a) It produces this trace

```
c=-1 A,B,C,D,E,F,
c=0  F,B,C,D,E,A,
c=1  F,E,C,D,B,A,
c=2  F,E,D,C,B,A,
c=3  F,E,C,D,B,A,
c=4  F,B,C,D,E,A,
c=5  A,B,C,D,E,F,
```

b) Nothing is achieved overall by the function as the array elements return to their original positions.

c) Half way through achieves: ARRAY REVERSE

d) arrayReverse included in code below

```
#include "stdio.h"; //to use printf
int a[6]={'A','B','C','D','E','F'}; //initialise array a

void dump(c){ //added code to simulate trace
	int i;
	printf("c=%d ",c);
	for(i=0;i<=5;i++)printf("%c,",a[i]);
	printf("\n");
}

void f(b){
	int c,d;
	dump(-1); //trace
	for(c=0;c<=b;c++){
		d=a[c]; a[c]=a[b-c]; a[b-c]=d;
		dump(c); //trace
	}
}

void arrayReverse(int last){
	int i,swap;
	for(i=0;i<=(int)last/2;i++){ //only go half way
		printf("arrayReverse i=%d\n",i);
		swap=a[i]; a[i]=a[last-i]; a[last-i]=swap;
	}
}

int main(){
	f(5);
```

```
	arrayReverse(5);
	dump(9);
}

//f as a whole is a no-op - does not achieve anything
//half of f is equivalent to reverse or arrayReverse
```

Examiners' Guidance Notes

This was a popular question attempted by three-quarters of the candidates, although only a quarter reached the required standard. Many candidates traced the function successfully; typical errors were to assume the function simply swapped the array elements to put them in reverse order.

Few candidates identified that the array was reversed half-way through the trace consequently they were unable to make a reasonable attempt at part d).

A4

Answer pointers

a)

```
int f(int i,char c){
	int j=i;
	if(c>='A'&&c<='Z')
		j++;
	return j;
}
```

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

c) If any of the 5 spaces were removed the program would give compilation errors. For example, if the first space were omitted the compiler would see intf as a single "word"/token and not two "words"/tokens – and the code would not form a valid program.

d)

```
identifiers(4): f,i,c,j
constants(2): 'A', 'Z'
operators(4): =, >=, <=, ++
longest Boolean expression: c>='A'&&c<='Z'
conditional statement:  if(c>='A'&&c<='Z')j++;
```

e)

```
int f(int i,char c){
	return ((c>='A'&&c<='Z')?j++:j); //neither set of () is actually
needed
}
```

Examiners' Guidance 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 for not indenting their code. The majority of candidates gained high marks in parts b) and c) regarding readability and the use of space in programs.

Surprisingly few candidates were able provide a correct conditional statement, a conditional expression or to identify the different operators used in the example code.

Most candidates failed to attempt part e) and those who did generally misunderstood the requirements of the conditional expression identified in the question.

SECTION B

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

B5

Answer pointers

a) Pseudocode expression below.

```
Variable Declarations
        Float BMI, H, W
        BEGIN
                BMI ← W / (H^2);
        END
```

b) Accept alternative solutions in pseudocode or program code

```
Variable Declarations
        Float BMI, H, W
        String Category
        INPUT "Height" H
        INPUT "Weight" W
        BEGIN
                BMI ← W / (H^2);
                IF (BMI <= 18.5) THEN
                        Category ← "Underweight"
                ELSE IF (BMI <= 25.0) THEN
                        Category ← "Normal"
                ELSE IF (BMI <= 30) THEN
                        Category ← "Overweight"
                ELSE
                        Category ← "Obese"
                END IF
                PRINT "Body Mass Index = "BMI;
                PRINT "Health Category = "Category;
        END
```

Examiners' Guidance Notes

Approximately half the candidates selected this question, around three-quarters of whom met the required standard. Part a) was generally answered well. Many candidates lost marks in Part b) by failing to include the BMI calculation or omitting to output the BMI value.

B6

Answer pointers

a) Pseudocode Example (accept any valid alternative or programming solution)

```
Pseudocode Convert Linear Search
Variable Declarations

INTEGER ARRAY Data_Items[15] = {1,54,4,76,32,12,14,3,31,52,65,45,13,89,17}
                                                    /* Data Item Array
INTEGER Item                                        /* Search Item
BOOLEAN Found                                       /* Item Found True/False

BEGIN

    PRINT "Enter Search Item"
    INPUT (Item)
    Found ← False

    FOR (i=1 TO 15)
        IF (Item == Data_Items[i]) THEN
            PRINT "Item Found" Item "At Array Location" i
            Found ← True
            Break                                   /* Item Found so break from loop
        END IF
    NEXT

    IF (Found == False) THEN
        PRINT "Item does not exist in array"
    END IF

END
```

b) A binary search finds the location of a key value k, as follows:

1. Look at the value stored half way down the array, compare with k
2. If equal to k, search completed so stop
3. If k is smaller throw away top half of array, otherwise throw away bottom half
4. Repeat this process until search successful.

A binary search is not suitable for this particular data as it is not ordered.

Examiners' Guidance Notes

A number of answers for Part a) appeared to be the standard linear search method but without the array location being output. Many candidates omitted Part a) altogether, confining their answers to Part b) only.

Part b) was answered well by the majority of those who attempted it.

B7

Answer pointers

a)

i) Conditional execution: applying a test and, depending on the result, executing a particular group of statements (or not).

ii) Looping: (thinking of a pre-check loop) applying a test to see whether a particular group of statements should be executed or not then, if the statements are executed, going back and re-evaluating the condition to see if the statements should be executed again. In this way the statements may be executed any number of times (including none) until the condition causes the loop to terminate.

b)

i) Structure of conditional statement

if ([expression]) [statement]

Five components in strict sequence

token 'if', token '(', expression structure, token ')', statement structure

ii) Structure of loop statement

while ([expression]) [statement]

Five components in strict sequence

token 'while', token '(', expression structure, token ')', statement structure

Examiners' Guidance Notes

For part a) higher marks were given to candidates who provided a complete definition. Some gave very limited answers, such as "it's the IF statement", even though the coding examples they provided in Part b) indicated they had the required knowledge and understanding.

B8

Answer pointers

a) 390 records held in sequential file.

Number of key comparisons required are: minimum 1, maximum 390, average 195.

b)

i. 26 letters in groups of 2 will create 13 groups in index.
Each group in index will contain the same number of keys
Number of key comparisons required are: minimum 1, maximum 13, average 7

ii. 390 employees in 13 groups would result in 30 employees per group
Number of key comparisons in main file group: minimum 1, maximum 30, average 15

Examiners' Guidance Notes

This was the least popular question. It was attempted by very few candidates, 7% of whom met the required standard.

Although Part a) was an easy question, few managed to gain more than one of the four available marks. Part b) was not answered well and many omitted this part altogether.

B9

Answer pointers

a) Sketch showing the web page divided as described in the question.
This can be achieved using a frameset consisting of 5 frames

b) Typical use for the five parts of the web page could be:

Top strip - banner: logo & organisation name
Left-hand strip - navigation deeper into site
Central part – the main window - navigation controls change the content of this window.
Right-hand strip - advertising or useful links to other websites
Bottom strip - navigation (site map)

c) This web page structure can also be achieved as a single page by using tables or inline frames. This approach might not be favoured because every page has to contain and re-render the banner

[Note students may answer a) and c) in reverse in which case the negative point for frames is the problem with bookmarks and back buttons]

Examiners' Guidance Notes

Approximately 50% of the candidates selected this question, 70% of whom met the required standard. Part a) was generally answered well, although many candidates failed to explain how the layout could be achieved. Part b) was answered less well, with some candidates omitting this part of the question. Most candidates failed to attempt Part c) and those who did generally misunderstood the requirements of the question.

B10

Answer pointers

a) Algorithm - an effective method in which a list of well-defined instructions for completing a task will, when given an initial state, proceed through a well-defined series of successive states, eventually terminating in an end-state

b) Iteration - the process of repeating a given action - either a certain number of times or until a given condition is met

c) GUI - graphical user interface, nowadays most often WIMP (windows, icons, menus, pointing device)

d) Code generator - a software tool that can generate code for particular application areas from a very high level specification, without the need for a human to supply e.g. an algorithm

Examiners' Guidance Notes

This was the most popular question, selected by 85% of the candidates.

Algorithm: Surprisingly, many candidates were unable to provide a definition; some stated it was a programming language or confined their answer to an example.

Iteration: This was better understood, but only the better candidates mentioned how limits can be set on the process.

GUI: Most candidates gained high marks for this part of the question.

Code generator: Rather than provide a definition, a significant number of candidates just provided an example of software that contained some code generation facility.

B11

Answer pointers

Part	Line	Error Explanation	When Found
a	8	"d" not declared	compile time
b	7	index not allowed in b[1] as b is not array	compile time
c	13	last time round loop, b is 9, b+1 is 10, so a[b+1] is invalid	run time
d	9	")" missing	compile time
e	6	variable required (in place of 2)	compile time
f	11	boolean expression required after if	compile time

Examiners' Guidance Notes

To gain the required two marks for each error, candidates were required to state the line number of the code in error, an explanation of the error, and whether the error would occur at compile time or run-time. Few candidates managed to meet all requirements for all error types and only a very small number gained the full marks available. As mentioned in the General Comments above, candidates lost marks needlessly by not meeting the requirements of the question.

B12

Answer pointers

a) According to some the waterfall method has 5 Phases: Requirements, Specification, Design, Implementation and Testing

   According to others the waterfall method has 7 Phases: Feasibility, Analysis/Requirement, Design, Coding, Implementation Testing and Maintenance/Review

b) Accept any valid description of a phase of the waterfall method; some examples given below:
   - Requirements: gathering information about the behaviour of the system
   - Specification: writing down the required behaviour and getting agreement of customer
   - Design: choosing effective data structures and algorithms
   - Implementation: writing final code solution
   - Testing: choosing a deliberate testing strategy, for example, by module, white-box, black-box, unit testing, system testing.

Examiners' Guidance Notes

This was a popular question and many candidates gained high marks. Part a) was usually answered well and the examiners gave considerable latitude regarding the different wording and textbook interpretations of the various waterfall methods.

Part b) required a short description of three of the phases. Answers that were little more than the words used to name the phases did not gain high marks. What was required was a paragraph containing a concise description of the main contents.