BCS HIGHER EDUCATION QUALIFICATIONS
Level 4 Certificate in IT

April 2013

EXAMINERS' REPORT

Software Development

General comments on candidates' performance

The standard for this examination was higher than in previous sittings. However, although there were some excellent papers, a number of candidates lost marks by providing answers which contained minimal content or were unrelated to the question. Often the candidate seemingly had the required knowledge and understanding but provided insufficient detail to achieve a pass.

Many candidates did not read the questions properly, resulting in answers that were not wholly relevant to the question set, and gained few, if any, marks. Candidates are advised to spend time reading and understanding the question before writing their answers.

For questions involving code or lists of instructions, candidates lost marks through careless mistakes, or by failing to provide adequate comments or explanations. 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.

SECTION A

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

A1.
Answer pointers

(a)

```
found=0;
for(i=0;i<10;i++)
      if(A[i]==V)
            found=1;
```

(7 marks)

(b)

```
int indexOf(int V, int* A){
      int i;
      for(i=0;i<10;i++)
            if(A[i]==V)
                  return(i);
      return(-1);
}
```

(7 marks)

(c)

```
/*indexFrom has an extra parameter - and index to start looking
from*/
#include<stdio.h>
int indexFrom(int V, int* A, int from){
      int i;
      for(i=from;i<10;i++)
            if(A[i]==V)
                  return(i);
      return(-1);
}
int allDifferent(int* A){
      int i;
    for(i=0;i<99;i++) /*try each member of A */
            if(indexFrom(A[i],A,i+1)>=0) /* is there is a repeat to
the right */
                  return(0); /* if so, the values in A are not all
different */
      return(1);
}
```

(8 marks)

(d)

```
void main(){
      int A[100];
      int i;
      for(i=0;i<100;i++)
            scanf("%d",&A[i]);
      if(allDifferent(A))
            printf("The data values are all different");
      else
            printf("There is at least one repeated value in the
data");
}
```

(8 marks)

Examiners' Guidance Notes

Not many candidates attempted this question; in parts (a) and (b) the majority of candidates produced correct or near correct solutions for detecting if the value V is found in array A.

A few candidates created a working solution for part (c) where they used the extra index parameter that was needed in the "alldifferent" function. However the complete program for part (d) was well answered and candidates demonstrated the ability to read the data, use the previously designed functions and output an appropriate message.

A2.
Answer pointers

It would be important to check that the answer
a) contains a way of recognising the double zero
b) process all the Message array up to the double 0
c) counts the number of (identical) keypresses up to a zero
d) applies two subscripts to the Keypad array
e) that one is a value taken from the message array
f) the other is a count of identical keypresses
g) the 2 subscripts are in the right order
h) the letters of the message are written out (one by one as discovered) or all at once at the end
i) used sensible variable names
j) that were init`ialised properly
k) used a few well chosen comments
l) used a function

```
#include<stdio.h>
void decode(int* Message,char Keypad[10][4]){
      int lastKey=-1;
      int i=0;
      int count=0;
      int key=Message[i]; /* find first key pressed */
      while(key!=0 && lastKey!=0){ /* stop on 2 adjacent zeros */
            if(key==0) /* the end of a repeated sequence of identical
keys */
                  printf("%c",Keypad[lastKey][count]); /*decode*/
            else
                  count++; /*continue counting identical keypresses*/
            lastKey=key; /* current key moved to last key */
            i++;        /* and move ahead */
            key=Message[i]; /* to next key, then round again */
      }
}
void main(){
      char Keypad[10][4]; /* contains letter equivalents for keys */
      int Message[20]; /* contains key values 0-9 */
       /* setup Message */
       /* setup Keypad */
      decode(Message,Keypad);
}
```

Examiners' Guidance Notes

This was the least popular answer in section A and the few candidates that attempted it generally produced poor quality answers

Many missed the point of using a While Loop until a double zero input was detected to signify the end of message. Others attempted code that would only work for the "HI MUM" example that was given in the question.

Several candidates struggled to satisfactorily use a 2-dim array for the keypad even though an example of it was provided in the question.

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

A3.
Answer pointers

The array A has 10 elements and following code has been written to shift every value in array A one place to the right

```
void rshift(int* A){
      int i;
      for(i=0;i<10;i++)
            A[i+1]=A[i];
}
```

	0	1	2	3	4	5	6	7	8	9
A	'Q'	'W'	'E'	'R'	'T'	'Y'	'U'	'I'	'O'	'P'

a) Trace the call of rshift with the values of A shown in the table above

```
i     Result in A

0     'Q' in A[0] copied to replace 'W' in A[1]
1     'Q' in A[1] copied to replace 'E' in A[2]
...
8     'Q' in A[8] copied to replace 'P' in A[9]
9     'Q' in A[9] copied to ??? there is no A[10]
```

b) Describe in your own words the two main mistakes in this code

i) Code fills the array with Qs instead of shifting the individual letters one place right
ii) The code tries to access an element beyond the end of the array

c) Give a version of rshift which corrects these errors

```
void rshift(int* A){ /* start at the other end */
      int i;
      for(i=8;i>=0;i--)
            A[i+1]=A[i];
}
```

d) Modify your answer to (C) so that the value which would be 'lost' from the right is placed in the newly empty place on the left (this is called a cyclic shift to the right) - call this function rshiftc

```
void rshiftc(int* A){ /* start at the other end */
      int i, save=A[9];
      for(i=8;i>=0;i--)
            A[i+1]=A[i];
      A[0]=save;
}
```

e) Finally give a version of lshiftc which again is a cyclic shift but this time the values shift leftwards.

```
void lshiftc(int* A){ /* left shift */
      int i, save=A[0];
      for(i=0;i<9;i++)
            A[i]=A[i+1];
      A[9]=save;
}
```

Examiners' Guidance Notes

This was a popular question and in a few cases candidates gained maximum marks; many candidates gained high marks by correctly tracing the rshift function and identifying the two mistakes within it.

Few candidates were able to correct the rshift errors by decrementing the loop from the other end as shown in part (c) answer pointers above. Also in part (d) many candidates attempted to solve the problem of saving the 9^{th} element to the 0^{th} element by incorrectly swapping all of the elements within the FOR loop.

A4.
Answer pointers

```
a)

int f(int k){
      int i=0;
      while(i<k)
            if(A[i]<'E')
                  return(i);
      return(-1);
}

b)

"at any point" - is wrong because putting in spaces is only allowed
BETWEEN tokens, so for example putting a space into a reserved word
( wh ile ) is not allowed.

"all spaces out...remains the same" - is wrong because taking spaces
out of strings will change the meaning of the program ("the answer
is" is different from "theansweris")


c) Referring to the code in part (a), find and write out the
following
      i) all the different identifiers
            f, k, i, A
      ii) all the different constants
            0, 'E', -1
      iii) a conditional (logical, boolean) expression
            i<k
      iv) a conditional statement
            if(A[i]<'E')
                  return(i);
      v) the statement that is repeated by the loop
            if(A[i]<'E')
                  return(i);

d) Describe the structure of a for-loop (a counted loop) and
describe the way that it is executed.
      for(exp1;exp2;exp3)statement;
is executed as
      exp1;
      while(exp2){statement;exp3}
[In other words
a) exp1 is executed as an initialisation.
b) if exp2 is false the loop finishes.
if exp2 is true the 'statement' is executed followed by the 'step'
expression exp3, then there is a jump back to (b)
```

Examiners' Guidance Notes

Thus was a popular question that was answered by nearly all of the candidates the majority of whom passed.

The FOR loop was generally well described and the majority of candidates gained maximum marks for restructuring code into a more readable form. However, only a few candidates detected and corrected the errors in the free format programming definitions given in part (b) of the question.

Surprisingly few candidates were able provide a conditional statement or identify all the constants used in the code.

SECTION B

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

B5.

Answer pointers

(a) Pseudocode expression below.

```
Variable Declarations
        int t, n, Power
        float Final_Amount, P,r
        ..
        BEGIN
                Power ← n*t;
                Final_Amount ← P*((1 + r/n)^Power);
        END
```

(b) Accept pseudocode or program

```
Variable Declarations
        int t, n, i, Power
        float Final_Amount, P,r
        INPUT "Number of Years:" t
        INPUT "Annual Interest:" r
        INPUT "Number of Times Compounded Per Year:" n
        INPUT "Initial Investment:" P
        BEGIN
                i = 1
                WHILE (i < t ) DO
                        Power ← n*t;
                        Final_Amount ← P*((1 + r/n)^Power);
                        PRINT "Final Amount = £ "Final_Amount;
                        PRINT "Year = " i ;
                        Print newline
                        i++
                END WHILE LOOP
        END
```

Examiners' Guidance Notes

Approximately 20% of the candidates selected this question and around half of those met the required standard. Part a) was generally answered well, although many candidates confined their answer to only this part of the question. Part b) was answered less well, common errors being careless use of variables, incorrect use of the formula and failure to print the final amount.

B6.

Answer pointers

a) A search algorithm is used to find an item with specific properties amongst a group of items, whereas a sort algorithm is used to put elements from a list into a specific order.

b) Binary Chop Algorithm / Pseudocode

```
Binary-Search (A, Target):          //A is sorted array
                                    //Target is value being searched for
Variable Declarations
    Left ← 1                        // Set up initial conditions
    Right ← Size[A]                 //Right assigned to maximum element in array
    WHILE (Left <= Right) DO
    BEGIN
        Middle ← (Left + Right) / 2
        IF (A[Middle] = Target) THEN
            PRINT "Target Value = " Target "Found at A[" Middle "]"
        ELSE IF (A[Middle] < Target) THEN
            Left ← Middle + 1
        ELSE
            Right ← Middle – 1
        END IF
    END WHILE LOOP
```

Word descriptions accepted, such as:

- Array information must be ordered according to some key
- Envisage the keys in ascending order in an array
- To find key k look at the value stored half way down the array, compare with k
- If equal to k, search completed so stop
- If k is smaller throw away top half of array, otherwise throw away bottom half
- Repeat from step 3 until search successful or only one key left

c) In a linear or sequential search, each element in the list is examined until the target value is found; this could take considerable time for a large array. In a binary search the number of elements being examined is halved for each iteration of the program; for example a maximum of only 6 comparisons are needed to find a target value in a list of 64 elements.

Examiners' Guidance Notes

Part a) was answered correctly by most of the candidates.

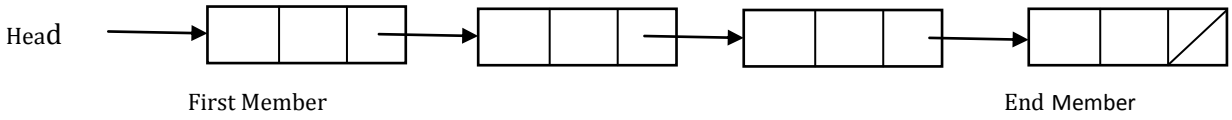
Although Part b) required the candidates to describe a binary chop search, some provided code or descriptions for a bubble sort and therefore failed to gain any marks. Most candidates appeared to understand the concept of a binary chop search but lost some marks by forgetting to describe the final iteration, assuming that the examiner had, by that stage, understood the process and that no further explanation was needed.

Part c) was answered well, with most candidates gaining at least one of the two available marks.

B7.

Answer pointers

a) Linked list diagram



b) Linked List Data Structure using C++

```
Struct ListNode
{
        String Name;
        Int Number;
        ListNode *Link;
};
```

c)

```
Variable Declarations
..
BEGIN
        OUTPUT "Enter Membership Number";          // Number request
        INPUT Target;                                      // Target is value being searched for

        Pointer ← Head                                  // Set pointer to head at start
        IF (Pointer = Null) THEN                        // Check if list empty
                PRINT "List Empty – unable to find membership number";
        ELSE
                WHILE ((Pointer not at node containing Target) AND (Pointer ≠ Null)) DO

                        Pointer ← Next Node             // Check next node

                        IF (Pointer at node containing Target) THEN          //Target found
                                PRINT "Name:"  Name  "Number:"  Target;

                        IF (Pointer = Null) THEN                                    //Target not in List
                                PRINT "unable to find membership number";

                END WHILE LOOP
END
```

Examiners' Guidance Notes

This was the least popular question in Section B and candidates generally did not perform well.

Part a) was moderately well answered, although candidates lost marks by showing only one data item or more than the required four nodes.

Part b) was not answered well, with many omitting this part of the question.

Although a straightforward question, many candidates either omitted Part c) or failed to carry out the instruction to output the name and membership number.

B8.

Answer pointers

a) Why is documentation important? – The team who wrote software are not necessarily the same team that will maintain or modify the software.
b) Who is it for? – Documentation is for the users and follow-on programmers
c) What does it consist of? - Separate documents, comments in code
d) How is it produced? - By hand sometimes, automatically sometimes
e) When is it produced? - As part of the process it describes/records
f) What are the problems with it? – Documentation is often inadequate or out-of-date

Examiners' Guidance Notes

This was a popular question. Candidates who read the question parts carefully were able to gain high marks. However, many ignored the six question parts and wrote an essay about documentation; generally those candidates failed to gain the high marks available. Other candidates lost marks by writing about general company documentation, such as annual reports and balance sheets; as these are not relevant to systems development, no marks were given.

For Part f), many candidates wrote about the advantages of documentation rather than the problems and no marks could be given for such answers. Candidates are advised always to read the question carefully before answering.

B9.

Answer pointers

3 marks per element for (identifying element, drawing element, describe suitable use).

Typical answers include:

- **Drop-down menu / list box/ combo box** (largely hidden apart from one choice until clicked on) - choose 1 from many or several from many
- **Radio buttons** (circles) - choose exactly 1 from many
- **Check boxes** (squares) - choose none, one or several from many
- **Text box** (rectangle, 1 line) - a little text (can be made secret for password)
- **Text area** (rectangle, multi-line) - a lot of text, such as address

Examiners' Guidance Notes

This was answered very well. Candidates clearly understood the requirements and provided the required identification, drawing and description of the elements. Although the question stated that it was unnecessary to draw a whole web page, many candidates did so; no marks were deducted but candidates probably lost valuable examination time

B10.

Answer pointers

a) Compiler/interpreter. A compiler translates a high level language program to low level language suitable for later running on a computer. An interpreter translates a high level program statement (and executes it) before moving on to the next statement.

b) Data structure/program structure. Data structure concerns the recognition of sequencing (record), selection (variant, union) and repetition (array, file) within data and recognizing primitive data types (i.e. not made up of other types) e.g. integer, character, Boolean…). Program structure is the recognition of sequencing (;), selection (if) and repetition (for, while) in programs.

c) Black box/white box testing. Black box testing is about testing a subroutine via its interface with no knowledge of its inner workings (code). White box testing is about testing a subroutine with full knowledge of its code.

d) Hardware/software. Hardware is the physical parts of a computer, such as keyboard, processor or printer. Software is concerned with sets of instructions or data for a computer which can be written down, but for execution have to be ultimately stored in sequences of binary digits encoded in some physical way.

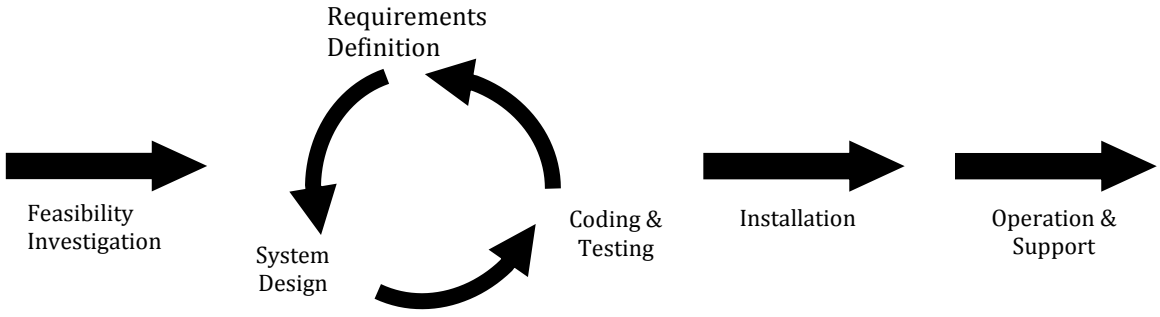
Examiners' Guidance Notes

This was a popular question that was generally answered well. Most candidates demonstrated the required knowledge, particularly in respect of Part c). However, many omitted Part b) altogether or merely provided answers containing the words "structure", "data" and "program" without any particular meaning being attached.

B11.

Answer pointers

a). The diagram below represents evolutionary prototyping.



Diagrams that represent throwaway prototyping, incremental prototyping or extreme prototyping are also acceptable.

b). Brief explanation for evolutionary prototyping should include the following points:

The method is similar to the traditional waterfall method except that the developer repeatedly loops through the Requirements, System Design and Coding until a satisfactory solution is obtained.

Explanations for throwaway prototyping, incremental prototyping or extreme prototyping are also acceptable.

c). TWO advantages described from the following:

- 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

Some good understanding was demonstrated, illustrating the candidates' own experiences. Although Part a) clearly required a drawing, many provided a description and gained limited marks. Part b) was generally answered well, even by those candidates who omitted Part a). Some candidates wasted time in Part c) by repeating points they had previously made.

B12.

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 and fix it. Debugging is the stage after checking that the syntax is correct.

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

Answers were disappointing. For Part a), many candidates omitted the fact that debugging includes the resolution of the error, not just the identification. For Part b), candidates frequently either repeated the comments made in Part a) or suggested that "an expert be employed", and gained no marks. Only the better candidates were able to answer Part c) and they generally had the required knowledge and understanding.