BCS THE CHARTERED INSTITUTE FOR IT

BCS HIGHER EDUCATION QUALIFICATIONS
BCS Level 4 Certificate in IT

SOFTWARE DEVELOPMENT

September 2016 - Morning
Time: TWO hours

Section A and Section B each carry 50% of the marks.
You are advised to spend about 1 hour on Section A (30 minutes per question)
and 1 hour on Section B (12 minutes per question).

Answer the Section A questions you attempt in Answer Book A
Answer the Section B questions you attempt in Answer Book B

The marks given in brackets are **indicative** of the weight given to each part of the question.

Calculators are **NOT** allowed in this examination.

General comments on candidates' performance

The standard for this examination was reasonable, with many candidates gaining high marks.

Although there were some excellent papers, there is evidence that a large number of candidates lost marks 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 structuring and writing their answers.

For questions involving code or lists of instructions, many candidates lost marks through careless mistakes or crossings out, making it difficult for the examiners to see which variables and commands were being used or to understand the program structure.

Note that the answer pointers contained in this report are examples only. Full marks were given for alternative valid answers.

SECTION A
Answer 2 questions (out of 4). Each question carries 30 marks.

A1

a) Given an array V containing experiment readings, write a function ***findIndex***() to find the index of the first run of 3 readings that are identical. For example in the array V below the first run of three identical values starts at index 2 so the result required is 2.

[Note: If a run of three is not found, the function should return -1]

index	0	1	2	3	4	5	6	7	8	9
V	3	9	7	7	7	1	5	5	5	5

(10 marks)

b) Incorporate your function into a complete program that prints not only the index of the run of three but the repeated value as well. So using the array V above output would be 2, 7.

(8 marks)

c) Write a function ***findRun***(n,r) that can take 2 parameters (n=the length of the array and the r=length of the run of identical values) and returns the index where the run is found. So, using the array V above, findRun(10,4) would return 6.

(12 marks)

Answer Pointers

a)
```
int findIndex(){
        int i;
        for(i=0;i<8;i++){ /* stop 2 short of length of array */
                if(V[i]==V[i+1] && V[i]==V[i+2]){
                        return i;
        }
        return -1;
}
```

b)
```
int V[]={3,9,7,7,7,1,5,5,5,5} /* not required from student*/
int findIndex(){
        :
}
void main(){
        var idx=findIndex()
        if(idx<0){
                printf("no run of 3 found");
        }else{
                printf("%d,%d",idx,V[idx]);
        }
}
```

c)
```
int findRun(int n, int r){
        int i;
        for(i=0;i<=n-r;i++){ /* stop r short of length of array */
                foundRun=1; /* true */
                for(j=1;j<r;j++){
                        if(V[i]!=V[i+j])
                                foundRun=0; /* false */
                }
```

```
            if (foundRun)
                return i;
        }

}
```

Examiners' Guidance Notes

Although this was not a popular question in section A, several of the candidates who attempted the question produced a correct or near correct solution for part a) writing a function findIndex to determine when 3 consecutive array elements are equal.

In part b) many candidates created an approximate solution by repeating the code from part (a) rather than calling the function and adding the appropriate print statements; candidates are advised to incorporate functions where possible to avoid making mistakes and simplifying the answer.

Few candidates gained marks for writing the findRun function, failing to incorporate the parameters (n) for the length of the array and (r) for the length of the run of identical values correctly.

A2

In a certain game played with cards, a player is dealt a hand of seven cards.

Each card has a value (Ace(one), Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten, Jack(eleven), Queen(twelve) or King(thirteen)) and each card has a suit (Clubs(one), Diamonds(two), Hearts(three) or Spades(four)). A "run of three" requires a consecutive sequence of three values from the value list (e.g. "Four, Five, Six" or "Ten, Jack, Queen") in the hand of seven cards. In addition a "run of three" requires the three cards to be all of the same suit. So for example Ten of Hearts, Jack of Hearts, Queen of Hearts is a "run of three" in the suit Hearts.

A single card can be represented by a pair of numbers (V,S). The first number V (in the range 1 to 13) represents the Value and the second number S (in the range 1 to 4) represents the Suit.

So a hand of seven cards can be represented by two arrays

Index	1	2	3	4	5	6	7
V	4	3	4	10	2	10	8
S	1	1	3	4	1	2	3

In this hand of seven cards, the "run of three" is (2,1) the "Two" of "Clubs", (3,1) the "Three" of "Clubs" and (4,1) the "Four" of "Clubs".

You should assume the two arrays V and S are already declared and set up with values.

a) Write a function, int ***InHand***(int v, int s), which returns 1 (TRUE) if a card of value *v* and suit *s* exists in the seven cards defined by the arrays V and S or 0 (FALSE) otherwise.

(10 marks)

b) Using the function ***InHand*** from part a), write a program to find out whether a given hand of seven cards contains a "run of three".

The output from the program should be either a message saying "Run of three not found" or a listing of the three cards in sequence (e.g. Two of Clubs, Three of Clubs, Four of Clubs).

(20 marks)

Answer Pointers

```
#include <stdio.h>
#include <string.h>
int V[]={0, 4, 3, 4,10, 2,10, 8}; /* Values */
int S[]={0, 2, 1, 3, 4, 1, 2, 3}; /* Suits */
int inHand(int v,int s){ /* is the card (v,s) in the hand? */
        /*printf("inHand %d,%d\n",v,s);*/
        int c;
        for(c=1;c<=7;c++){
          if(V[c]==v && S[c]==s)
                  return 1; /*true*/
        }
        return 0; /*false*/
}
void printCard(int v, int s){ /* convert number codes to printable strings */
        /*printf("printCard %d,%d\n",v,s);*/
        char valstr[10],suitstr[10];
        switch(v){
          case 1: strcpy(valstr,"Ace");break;
          case 2: strcpy(valstr,"Two");break;
          case 3: strcpy(valstr,"Three");break;
          case 4: strcpy(valstr,"Four");break;
          case 5: strcpy(valstr,"Five");break;
          case 6: strcpy(valstr,"Six");break;
          case 7: strcpy(valstr,"Seven");break;
          case 8: strcpy(valstr,"Eight");break;
          case 9: strcpy(valstr,"Nine");break;
          case 10: strcpy(valstr,"Ten");break;
          case 11: strcpy(valstr,"Jack");break;
          case 12: strcpy(valstr,"Queen");break;
          case 13: strcpy(valstr,"King");break;
        }
        switch(s){
          case 1: strcpy(suitstr,"Clubs");break;
          case 2: strcpy(suitstr,"Diamonds");break;
          case 3: strcpy(suitstr,"Hearts");break;
          case 4: strcpy(suitstr,"Spades");break;
        }
        printf("%s of %s\n",valstr,suitstr);
}
void findRunOf3(){ /* check each card as candidate for start of run */
        int c,foundStartVal=0,foundSuit=0;
        for(c=1;c<=7;c++){
          /*printf("findRunOf3 %d,%d,%d\n",c,foundStartVal,foundSuit);*/
          if(
                  inHand(V[c]+1,S[c])
                  &&
                  inHand(V[c]+2,S[c])
          ){
                  foundStartVal=V[c];
                  foundSuit=S[c];
          }
        }
        if(foundSuit==0)
          printf("Run of 3 not found");
        else {
          printCard(foundStartVal,foundSuit);
          printCard(foundStartVal+1,foundSuit);
```

```
            printCard(foundStartVal+2,foundSuit);
        }
}
int main(){
        findRunOf3();
}
```

Examiners' Guidance Notes

This was the least popular questions in Section A. There were some good attempts at writing the InHand function in part a) where candidates checked that each card in the seven card hand was a correct value and suit based on the array structure given.

The evidence shows that few candidates used the InHand function in part b) to check whether a hand had a run of three cards. The candidates also failed to convert the card number data into printable strings to output their results.

A3

a) Trace the behaviour of the call ***f***() when ***f*** and V are defined as shown below.

index	0	1	2	3	4	5
V	4	2	5	6	1	3

```
int f(){
        int g=V[0];
        for(var h=1;h<6;h++){
                g+=V[h];
        }
        float i=g/6;
        float j=V[0]-i;
        for(var k=1;k<6;k++){
                if(V[k]-i>j)
                        j=V[k]-i;
        }
        return j;
}
```

(14 marks)

b) Describe in a single sentence the overall effect of the function ***f***.

(6 marks)

c) Rewrite the function changing all the identifiers to make the code more readable and understandable and add suitable comments.

(10 marks)

Answer Pointers

a)

g=V[0]=4
h=1, g+=V[1], g=6
h=2, g+=V[2], g=11
h=3, g+=V[3], g=17
h=4, g+=V[4], g=18
h=5, g+=V[5], g=21 = Total

Average = Total / elements so i=21/6=3.5

First element – average so j = 4-3.5 = 0.5

k=1,	i-V[k]>j,	2-3.5 > - 0.5,	no	
k=2,		5-3.5 > 1.5	yes	j=1.5
k=3,		6-3.5 > 2.5	yes	j=2.5
k=4,		1-3.5 > -2.5	no	
k=5,		3-2.5 > 0.5	no	

return 2.5

(14 marks)

b) function computes the average of the array and then checks to see how far each number is above the average and reports the largest amount above the average

(6 marks)

c)

```
int largestAboveAverage(){    /* computes … */
        int total=V[0];
        for(var i=1;i<6;i++){    /* find total of array */
                total+=V[i];
        }
        float ave=total/6;        /* find average */
        float maxOver=V[0]-ave;        /* distance above average */
        for(var k=1;k<6;k++){
                if(V[k]-ave>j)
                        maxOver=V[k]-ave; /* save maximum so far */
        }
        return maxOver;
}
```

(10 marks)

Examiners' Guidance Notes

This was a very popular question with many candidates achieving high marks in part a) where they attempted to trace through the code provided. The evidence shows that in many cases all stages of the trace were completed, and the correct answer provided; although typical errors were that the total (g) and the average (i) were not calculated correctly.

The majority of candidates tabulated their trace which made it easy to follow. The evidence shows in some cases candidates simply wrote down a solution to the trace without showing how they arrived at it, consequently marks could not be awarded for method and technique if they produced an incorrect solution.

Many candidates gained full marks for part c) where they clearly understood the function of the code and chose meaningful identifiers and added appropriate comments to help explain the code

A4

a) Consider the code below and write it out in a more familiar human readable form.

```
void x(char y){int z=10;while(z--)printf("%c",y);}x('-');
```

(6 marks)

b) By adding to your answer for a) or writing out again, show where in the code there is

i) a function declaration
ii) a formal parameter
iii) a function call
iv) an actual parameter

(4 x 2 marks)

c) Referring to the code in part a), find and write out the following:

i) all the different identifiers
ii) all the different constants
iii) all the different operators
iv) an iterative (repetitive, loop) statement
v) the statement that is repeated by the loop

(5 x 2 marks)

d) Write out the code from a) again, this time replacing the loop with an equivalent for loop.

(6 marks)

Answer Pointers

a)

```
void x(char y){
        int z=10;
        while(z--)
                printf("%c",y);
}
x('-');
```

(6 marks)

b)

function declaration
        void x(char y){int z=10;while(z--)printf("%c",y);} (2 marks)
formal parameter
        y (2 marks)
function call
        x('-') (2 marks)
actual parameter
        '-' (2 marks)

c)

i) identifiers
        x, y, z (3 marks)
ii) constants
        10, "%c", '-' (3 marks)
iii) operators
        =, -- (2 marks)
iv) iterative statement
        while(z--)printf("%c",y); (1 mark)
v) repeated statement
        printf("%c",y); (1 mark)

d)

```
        int z=10;while(z--)printf("%c",y);
                becomes
        for(int z=10;z>0;z--)printf("%c",y);
```

(6 marks)

Examiners' Guidance Notes

This was a very popular question that was attempted by 94% of the candidates; it was generally well answered with approximately 70% 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 c).

The evidence shows that marks were mainly lost in part b) where many candidates found it difficult to identify the parameters used and the function declaration / function call.

Few candidates were able to replace the "while" loop with a "for" loop; typical errors were to omit the condition within this statement (z>0).

SECTION B

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

B5 Write a function Power (P, N) in pseudocode (or a program in a language of your choice) in which the integer value of 'P' is raised to the integer power of 'N'.

a) Using recursion (6 marks)

b) Using iteration (6 marks)

Answer Pointers

a) Recursion - accept pseudocode or program

```
FUNCTION Power (P, N: INTEGER): INTEGER
        {Using Recursion}
        IF (N = 0)
                Power ← 1;
        ELSE
                Power ← (P * Power (P, N-1);
        END
END FUNCTION
```

b) Iteration - accept pseudocode or program

```
FUNCTION Power (P, N: INTEGER): INTEGER
        {Using Iteration}
        INTEGER Temp, X
        IF (N = 0)
                Power ← 1;
        ELSE
        BEGIN
                Temp ← 1;
                FOR (X=1; X<=N; X++)
                        Temp ← Temp * P;
                END
                Power ← Temp;
        END
END FUNCTION
```

Examiners' Guidance Notes

This was an unpopular question with only around 15% attempts; those candidates who chose it did not perform well compared to other questions in Section B. Most candidates elected to use a programming language rather than pseudocode. The evidence shows that the code that was produced showed a lack of understanding of recursion in general. Much of the code produced was often let down (and hence marks lost) by carelessness such as variables were not declared; or values were not assigned to these variables; the test for N=0 not present.

Overall answers to part b) were better than those to part a) but nearly all candidates attempted both parts with varying success. There were some candidates who demonstrated a lack of coding experience and knowledge of recursion elected to simply calculate the power of n in a single assignment statement.

B6 A palindrome is a word, or an array of characters, that reads the same backwards as forwards, such as: nurses or radar.

a) Write a pseudocode function "palindrome" which takes as its parameters a character array and which returns "TRUE" if the array contains a palindrome otherwise returns "FALSE". (8 marks)

b) Trace through the code for the inputs: "rotator" and "notion". (4 marks)

Answer Pointers

a) Accept pseudocode, including a range of alternative approaches

```
FUNCTION Palindrome (Input: STRING): BOOLEAN
        INTEGER Len, X
        Palindrome ← TRUE;
        Len ← Input.length;
        /* Loop through half the length of the string for full comparison */
        FOR (X=0; X< Len / 2; X++)
                /* Check the first character, doesn't equal the last character and work to centre*/
                IF (Input[X] != Input[Len –X] )
                        /* Not a palindrome so break as no need to check further*/
                        Palindrome ← FALSE;
                        Break;
                END
        END
        Return Palindrome;
END FUNCTION
```

b) Trace table

Input = ROTATOR Length = 7 Len/2=3

X	Input[X]	Input[7 - X]	Palindrome
0	R	R	TRUE
1	O	O	TRUE
2	T	T	TRUE
3	A	A	TRUE

Input = NOTION Length = 6 Len/2=3

X	Input[X]	Input[7 - X]	Palindrome
0	N	N	TRUE
1	O	O	TRUE
2	T	I	FALSE

Examiners' Guidance Notes

This was an unpopular question with just over 11% attempting it. However, about 25% of candidates either gained most of the marks available or provided solutions that did not address the question at all. The evidence shows that some candidates were unable to provide suitably clear solutions to part b) indicating that they had not fully applied trace tables to test input conditions.

Candidates were expected to derive general purpose pseudocode from the definition of a palindrome. Marks were lost if candidates used the two strings provided to test their algorithm rather than allow any strings to be tested. In these answers the two strings provided were present in the pseudocode. There were also some 'economical' solutions; for example a few candidates used a function that reversed the string so that two strings were compared rather than iterate through the array character at a time. Although this answers the problem candidates are advised not to unduly rely on shortcuts such as built in functions may not be available or suitable.

B7 Consider the function min (see box below).

a) How would a black-box test of the function min be performed? **(3 marks)**

b) How would a white-box test of the function min be performed? **(3 marks)**

c) In what way would the error in the function show up under black-box testing using the test case min(4,2,3)? **(3 marks)**

d) In what way would the error show up under white-box testing using the test case min(4,2,3)? **(3 marks)**

```
int min(int a, b, c){
/* specification: find the minimum of 3 values given as parameters */
int min; /* hold the minimum value found so far */
        min=a;
        if(b<a) min=b;
        if(c<a) min=c;
return(min);
        }
```

Answer Pointers

(a) Black-box testing takes place with an executable version of the module *min*. Test sets (perhaps with the minimum value in each of the three possible positions) are created and the module is run with test case input and the results inspected to see if they match with the expected results (obtained from the specification)

(b) White-box testing takes place with the source code of *min* available. Test sets are created and all paths through *min* are subjected to a dry run to see that the code performs as expected.

(c) Under black-box testing the result of running min(4,2,3) would be 3. The expected answer is 2 as defined by the specification of the function. Therefore, the actual and expected results do not agree and an error has been found.

(d) [The comment about the purpose of the local variable min is critical in this part of the question] Under white-box testing the function first considers a and assigns the value of 4 to min - so min can be said to contain the minimum of all the values considered so far - just 4.

The first 'if' statement results in the assignment of 2 to min. So, min can be said to contain the minimum of all the values considered so far (4,2)

The second 'if' statement results in the assignment of 3 to min. This new value of min is larger than the previous value and so it violates the meaning of the variable min as defined by the comment. So, the error is discovered on executing the second 'if' statement.

Examiners' Guidance Notes

A very popular question with almost 70% of candidates attempting it. The evidence shows that most candidates could differentiate black box from white box testing although many answers were too verbose and often included the answers to parts c) and d) mixed up in them. There was generally a good understanding of applying black box testing and less so for white box testing despite being directed by the comment in the question.

B8 a) Use an algorithm to describe the operation of a queue.

(10 marks)

b) A queue contains the values 7, 6, 9 where 7 is the first value stored and 9 is the last.

List the final values in the queue after the following operations are made:

1. Pop
2. Pop
3. Push12
4. Push 16
5. Pop
6. Push 11

(2 marks)

Answer Pointers

a) A queue is used to as a temporary storage space for data. It is defined as an abstract data structure which operates on a **first in, first out (FIFO)** basis. The queue uses a two pointer (index) to keep track of the data in the queue, where:

- The start pointer indicates the data item that was first entered into the queue
- The end pointer indicates the data item that was last entered into the queue.

Initially Queue is empty so Start pointer is set to 0 and Queue has “Max” elements

Queue Push Algorithm (data inserted into queue)

```
if start_pointer = 1 and end_pointer = MAX then
  return queue full message
else if start_pointer = end_pointer + 1 then
  return queue full message
else begin
  queue [end_pointer] = data;
  end_pointer = end_pointer + 1;
end
```

Queue Pop Algorithm (data removed from the queue)

```
if start_pointer = 1 then
  return queue empty message
else begin
  data = queue[start_pointer];
  start_pointer = start_pointer + 1
end
```

b) Final content of queue 12, 16, 11

Examiners’ Guidance Notes

Almost half of candidates attempted this question but less than a half of these candidates gained sufficient marks for a pass. This question produced the lowest average mark on Section B. The reasons are not clear, but candidates should reflect on the comments that follow.

A third of the marks for part a) were available for candidates who knew that a queue was a FIFO data structure which supported the operations push and pop. There is evidence that some candidates thought that a queue was a LIFO data structure.

The evidence shows that just over half of the candidates who attempted part a) were generally able to give acceptable algorithms for push and pop although many candidates did not deal with the state of the queue being empty and full.
Most candidates generally answered part b) by writing down three numbers without any working. If the three numbers were ordered correctly then full marks were gained. However, credit was given for incorrect answers if candidates had shown how they arrived at their answer. Therefore, candidates are advised to show working out and demonstrate their knowledge of process even if that process is inaccurately applied.

B9 A business selling to the general public via a web site will require that site to have a good user interface design.

a) List the basic web elements that could be used with which the user can interact / enter data (6 marks)

b) Briefly describe what you consider to be the basic concepts of good user-interface design? (6 marks)

Answer pointers

a) Basic elements include:

- FORM
- MENU / SELECT
- LIST BOX
- TEXT BOX
- TEXT AREA
- BUTTON (SUBMIT / RADIO / CHECKBOX)

b) No standard answer; typical phrase / comments include:

- Simple, uncluttered
- Consistent layout and operation
- Can see what to do next
- Don't strain user's short term memory
- Design for usability
- Can undo or get back from a wrong move
- Text rollover to explain web element functionality
- On-screen help / Live chat

Examiners' Guidance Notes

This was the most popular question in this Section, with around 90% of candidates attempting it. This question was well answered overall and there was a wide range of acceptable answers. Many candidates gained close to full marks. But the evidence shows that a small number of candidates diverted from what the question asked. Also, some candidates may have misunderstood the meaning of the keyword 'elements'. A common diversion in candidate answers was to discuss the general principles of web page design rather than a discussion of form elements supporting I/O so these candidates gained few marks. Similarly, some candidates mixed up both parts of the question by depicting a screen design without any comments and therefore these candidates did not answer any part of the question and lost marks.

B10 Compare the following pairs of terms.

a) Source code and object code **(4 marks)**

b) Text file and binary files **(4 marks)**

c) Open and closed source software **(4 marks)**

Answer Pointers

a) Source code is the language instructions that have been written by the computer programmer. The computer cannot run or execute the source code directly whereas object code (or executable code) is translated from the source code using an assembler or compiler into numerical instructions that can be understood and executed by the computer.

b) Text files are used to store ASCII data and so they are human-readable using a text editor whereas binary files are used to store executable programs and numeric data files containing unprintable characters that are only computer-readable.

c) Open source software is freely available for download from the internet; the user is permitted to copy, edit and change the software as required. Whereas closed source software is licensed to the purchaser and gives the right to use the software, but not to modify it or redistribute it. The software developer holds the copyright to the software product

Examiners' Guidance Notes

This was a popular question with around 80% of candidates attempting it. This question produced the highest number of successful answers with around 70% gaining a pass mark.

The evidence shows that marks were generally lost in misunderstanding the nature of object code either mixing up object code with binary files even though they may both share encodings and stored as files. But they have different roles and the contexts as the question suggests. There was a very good understanding overall of open source versus closed source software with good examples given to reinforce candidates' answers.

B 11

a) The Rapid Application Development (RAD) methodology was developed in response of the need to deliver systems fast. Describe how the following management procedures are used support this approach:

i) Prototyping
ii) Incremental Development
iii) Time-Boxing

(6 marks)

b) Discuss the benefits and limitations of adopting the RAD methodology to develop software.

(6 marks)

Answer Pointers

a) **Prototyping** is an iterative approach based on producing a working model as fast as possible. The prototype is then refined using feedback from the software developers as well as the subsequent users of the system.

Incremental development based on refinement, is the hallmark of this methodology in which prototyping and iteration go hand in hand; in this case the throwaway prototype approach is refined to create production software

Time-boxing is a time management technique that is focussed on completing each task within a fixed time span. Consequently the development team will do their best to get a solution within that time frame, although in some cases this will be at the expense of functionality

b) Discussion points might include but not limited to:

Benefits

- The production version of an application is available earlier than using other methodologies like the Waterfall approach and therefore tends to cost less to produce
- There is a concentration on the essential elements needed for the user with an emphasis on fast completion

Limitations

- The emphasis on speed of development may impact the overall system quality

- Potential for inconsistent designs and a lack of attention to detail in respect of administration and documentation

Examiners' Guidance Notes

Another popular question with over 70% attempts. There was a good range of answers with most candidates, around 65%, gaining passes in this question. Part b) produced the most limited range of answers with many candidates producing long unstructured answers that failed to achieve the marks that were available.

B12

Consider the design and implementation of the search facility in a text-editor. During the process a specification will be drawn up, sometime later an algorithm will be produced and finally some documentation will be written. With reference to this example, write brief notes on what you understand by the following terms:

a) Specification **(4 marks)**

b) Algorithm **(4 marks)**

c) Documentation **(4 marks)**

Answer Pointers

a) A specification is a concise, complete description of what is required; preferably without hints at implementation [e.g. user should be able to specify a pattern to search for in the text, starting at the current cursor position]

b) An algorithm is a programming description of how the specification is to be achieved.
[e.g. If pattern is P and cursor is at initial position IC, set C to IC;
then look for pattern P at C and if not found move C one place forward and search again.
If C is beyond the end of file, start C at beginning of file. If C reaches IC again, stop as whole file searched.]

c) Documentation for the user would give information on how to use the search facility.
[e.g. which menu selection to use to start the editor, how to fill in the pattern text box, which button to press to start the search, continue the search, abandon the search, etc.]

Examiners' Guidance Notes

A popular question with around 76% attempts but the average mark was lower than expected for various reasons which are discussed below.

There is evidence of a large number of partial answers to this question where candidates demonstrated confidence that they could answer one or two of the parts but not all the question.
There were few fully correct answers to most parts of this question as candidates recalled and listed general principles of each topic. Candidates are advised that the most succinct way to answer similar questions is to give examples that relate to the text editor/scenario.

Many candidates thought that specification was a design stage in the SDLC and thought that an algorithm was entirely a coding/implementation process and not a design issue.