Software Development Techniques

SAMPLE TIME CONSTRAINED ASSESSMENT

MARKING SCHEME

This marking scheme has been prepared as a **guide only** to markers. This is not a set of model answers, or the exclusive answers to the questions, and there will frequently be alternative responses which will provide a valid answer. Markers are advised that, unless a question specifies that an answer be provided in a particular form, then an answer that is correct (factually or in practical terms) **must** be given the available marks.

If there is doubt as to the correctness of an answer, the relevant NCC Education materials should be the first authority.

Throughout the marking, please credit any valid alternative point.

Where markers award half marks in any part of a question, they should ensure that the total mark recorded for the question is rounded up to a whole mark.

Answer ALL questions

Marks

Question 1

a) A computer's *Central Processing Unit* (CPU) cannot directly execute instructions that are written in a language like a spoken language. Explain why this is the case. Give TWO (2) **types** of software tools that can create machine-code instructions. **4**

Mark scheme

CPU cannot directly execute instructions written in spoken language, i.e. high-level language (1 mark). It can execute low-level machine code only. (1 mark)

A compiler is a tool that can create low level CPU instructions, or binary machine code, from high-level computer language. (1 mark)
An assembler is another tool that can create low-level CPU instructions, or binary machine code, from assembly language. (1 mark)

****** Reject names of computer languages as types of software tools.***
****** Accept any other credible tool(s), e.g. translator, interpreter, etc.***

b) Identify **and** briefly explain THREE (3) qualities that a good algorithm should have. **6**

Mark scheme

1 mark for identifying, 1 mark for explaining, maximum 2 marks for any of the following (up to maximum 6 marks):

- ***It is complete: contains the steps required to arrive at the desired conclusion.***
- ***It is robust: can deal with unlikely situations and incorrect data.***
- ***It is efficient: accomplishes its task with the minimum number of steps.***
- ***It is readable: people can understand what it is doing.***
- ***It is maintainable: it can be updated/modified/changed easily.***
- ***It is documented: has instructions on how to make use of the algorithm.***

Total 10 Marks

Marks

Question 2

a) White-box testing, and black-box testing are two test strategies. Explain what they are **and** state which strategy desk-checking comes under. **5**

Mark scheme

White-box testing:

Any of the following or similar statements up to maximum 2 marks.

- ***Used to assess the flow of logic through a program (1 mark).***
- ***Makes sure the answer is correct from every branch of execution (1 mark).***
- ***Test cases are chosen to test all possible flows of execution through a program (1 mark).***

Black-box testing:

Any of the following or similar statements up to maximum 2 marks.

- ***Only considers the inputs and the outputs (1 mark).***
- ***Considers special cases where errors are particularly likely such as the boundary conditions (1 mark).***
- ***Does not look at the logic of the code within a function/program (1 mark).***
- ***Inputs are applied and the corresponding output results are recorded (1 mark).***
- ***It is all about seeing where inputs correctly match to outputs (1 mark).***

Desk-checking comes under white-box testing (1 mark).

b) *Regression testing* is an important method in making sure that programs are correct. Explain why regression testing is required **and** how it is carried out. Give TWO (2) examples of when regression testing is appropriate. **5**

Mark scheme

As changes are introduced to programs it is essential to prove that the changes have not introduced new flaws (1 mark). Regression testing uses a set of proven test cases (1 mark) that are applied each time a program is altered in some way (1 mark).

Examples of when regression testing is appropriate:

1 mark for each of the following, maximum 2 marks.

- **When program is enhanced with new code**
- **After any changes made during maintenance**
- **When program updated**
- **When problems/bugs fixed**

Total 10 Marks

Marks

Question 3

a) A clothes shop is counting their inventory of ladies' t-shirts. The following algorithm is used to count the number of small, medium and large sized t-shirts in a list of TEN (10) t-shirts stored in an array:

The t-shirt sizes in an array are as follows: [10, 12, 14, 8, 18, 12, 20, 8, 16, 14]

```
01. data sizes as array (10) of whole numbers
02. data counter as whole number
03. data large as whole number
04. data medium as whole number
05. data small as whole number
06. large = 0
07. medium = 0
08. small = 0
09. counter = 0
10. loop until counter is equal to 10
11.      if sizes[counter] is less than 12 then
12.          small = small + 1
13.      else
14.          If sizes[counter] is greater than 16 then
15.              large = large + 1
16.          else
17.              medium = medium + 1
18.          end if
19.      end if
20.      counter = counter + 1
21.  next loop
```

Marks

i) Copy the following table and fill in the values of ALL the data variables at line 19. The counter values are already filled in. You need to fill in all the cells of the table. When you finish, the counts of **small**, **medium** and **large** t-shirts should add up to 10. **10**

counter	*small*	*medium*	*large*
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			

Mark scheme

counter	*small*	*medium*	*large*
0	*1*	*0*	*0*
1	*1*	*1*	*0*
2	*1*	*2*	*0*
3	*2*	*2*	*0*
4	*2*	*2*	*1*
5	*2*	*3*	*1*
6	*2*	*3*	*2*
7	*3*	*3*	*2*
8	*3*	*4*	*2*
9	*3*	*5*	*2*

1 mark for each correct row (max. 10 marks)

All cells must be filled in for full marks. If some cells in columns are not filled in because they repeat, but the filled in values are correct, then deduct 1 mark from the total.

Total 10 Marks

Marks

Question 4

a) You are asked to write an application that will store daily weather information. You will need to decide on the data representation by using the most appropriate data types for the following variables: **temperature**, **name_of_month**, **day_of_month** and **is_dry_day** which is true if a dry day.

i) State what data types you will choose and explain why you would choose them. **8**

Mark scheme

1 mark for stating the type, 1 mark for explanation for each of the following:

- ***temperature: Real/float, as it will store a number that can include fractions.***
- **name_of_month*: String, as it will store text/alphanumeric characters only.***
- **day_of_month*: Whole number/integer, as it will store numbers without fractions.***
- ***is_dry_day: Boolean, as only one of the two possible values (true/false) can be stored.***

b) Data types can be *primitive* or *complex* types. Give ONE (1) example of each type. **2**

Mark scheme

Primitive types (value variables):

1 mark for any of the following, maximum 1 mark:

- ***Whole number/integer***
- ***Boolean***
- ***Real/float***
- ***Character***

Complex types (reference variables):

1 mark for any of the following, maximum 1 mark:

- ***String***
- ***Array of any type***

Total 10 Marks

Marks

Question 5

a) Loops are very useful in computer programs. There are two types: *bounded loop* and *unbounded loop*. Explain when you should use a *bounded loop* and when you should use an *unbounded loop*. **2**

Mark scheme

Bounded loop is used when it is known how many iterations the loop will do as it uses a counter (1 mark). Unbounded loop is used when it is not known how many iterations the loop will do as it is dependent on some condition (1 mark).

b) Below is a program that outputs the contents of an array. Re-write this program using a bounded loop. You do not need to declare the array. **7**

```
Output array[0]
Output array[1]
Output array[2]
Output array[3]
Output array[4]
Output array[5]
Output array[6]
Output array[7]
Output array[8]
Output array[9]
Output array[10]
Output array[11]
```

Mark scheme

```
data n as whole number
n = 0
loop until n = 12
   Output a[n]
   n = n + 1
next loop
```

Declaration and initialisation of index (2 marks), loop structure (2 marks), use of bounded loop type (1 mark), loop contents (2 marks).
***** Accept any credible alternative that has the above elements.**

c) Give ONE (1) **advantage** of using a loop in the program in b). **1**

Mark scheme

1 mark for each below, maximum 1 mark.

- ***The loop reduces the size of the program.***
- ***The program is easier to update/change if the array size is increased.***

Total 10 Marks

Marks

Question 6

a) Data structures are extensively used in computer programs. Two data structures employ access methods that are particularly useful: **First-in-first-out** (FIFO) and **Last-in-first-out** (LIFO). Give the names of the data structures and explain how their access methods work. **6**

Mark scheme

Queue (1 mark)
Stack (1 mark)

In Queue, new data is added to the end/tail of the queue (1 mark) and data is taken off the queue from the start/head of the queue (1 mark).
****** Reject usage of words like 'top' and 'bottom' for the queue.***
****** Reject explanation like 'first data in is the first data out'.***

In Stack, new data is added to the top of the stack (1 mark) and data is taken off the stack from the top of the stack (1 mark).
****** Reject usage of words like 'start' and 'end' for the stack.***
****** Reject explanation like 'last data in is the first data out'.***

b) Push and Pop are two operations that operate on a stack. Below are the two functions that implement the Push and Pop operations. The stack is represented by the **stack_array** and **si** is the stack index. These are globally declared.

i) The Push function is fine. However, the programmer made TWO (2) mistakes in the Pop function. Identify these mistakes and explain why they are mistakes. **4**

```
01. Function Push(needs d as whole number)
02.    stack_array[si] = d //put data d on top of stack
03.    si = si + 1         //increment stack index
04. End function

05. Function Pop returns whole number
06.    data d as whole number
07.    d = stack_array[si]
08.    si = si + 1
09.    return d
10. End function
```

Mark scheme

si = si + 1 (line 8) should be si = si – 1 (1 mark). This is required to point to the correct data on top of the stack (1 mark). si = si – 1 should then be between lines 6 and 7 (1 mark). The index must be pointing to the correct data on top of the stack <u>before</u> the data is accessed (1 mark).
****** Accept any alternative credible and correct explanation.***

Total 10 Marks

Marks

Question 7

a) Given the following logic expression:

(X and not Y) or (not X and Y)

i) Fill in the truth table below for this expression. T stands for True and F stands for False. **5**

X	Y	not X	not Y	C = (X and not Y)	D = (not X and Y)	C or D
F	F					
T	F					
F	T					
T	T					

Mark scheme

Truth Table

X	*Y*	*not X*	*not Y*	*C = (X and not Y)*	*D = (not X and Y)*	*C or D*
F	***F***	***T***	***T***	***F***	***F***	***F***
T	***F***	***F***	***T***	***T***	***F***	***T***
F	***T***	***T***	***F***	***F***	***T***	***T***
T	***T***	***F***	***F***	***F***	***F***	***F***

1 mark for each correct column (maximum 5 marks).
If column <u>not X</u> or/and column <u>not Y</u> is wrong, but rest are correct with respect to these then deduct up to 2 marks.
If column C or/and column D is wrong, but column <u>(C or D)</u> is correct with respect to these then deduct up to 2 marks.

ii) Comment on the values of **(C or D)** column with respect to the values in **X** and **Y** columns. **1**

Mark scheme

'(C or D) is True whenever X and Y are not equal', or alternatively, '(C or D) is False whenever X and Y are equal'. (1 mark)
***** Accept (C or D) is XOR/Exclusive OR logic**
***** Reject long comments that simply repeat what the table is showing.**

Marks

b) The following *if statement* assigns a value to the variable **B** depending on the value of the variable **A**. Both variables **A** and **B** are data type whole number.

```
if A > 0 then
   B = A + 1
otherwise
   if A < -1 and A > -5 then
      B = -1
   otherwise
      B = A
```

i) Fill in the missing values in the **B** column in the table below: **4**

A	B
8	
0	
-1	
-6	

Mark scheme

A	B
8	9
0	0
-1	-1
-6	-6

1 mark for each correct entry in the B column. (maximum 4 marks)

Total 10 Marks

Question 8

a) An array of whole numbers contains the numbers 2, 5, 10, 16, 28, 42, 55 in sorted order. How many binary search operations will be needed on the array before locating the number 55? Explain how you calculated this. **4**

Mark scheme

3 searches are needed (1 mark)

1. ***Locate the mid-point, which is 16. (1 mark)***
2. ***55 is greater than 16, so find mid-point of the numbers to the right of mid-point 16, which is 42. (1 mark)***
3. ***55 is greater than 42, and the next number is 55. (1 mark)***

****** Accept any alternative credible explanation.***

Marks

b) *Big O notation* gives us a generalised view of how well algorithms scale. Some typical *Big O notations* for search and sort algorithms are:

O(n)
O(log n)
O(n log n)

i) Identify which algorithms they correspond to. **3**

Mark scheme

1 mark for each one below (maximum 3 marks)
O(n): Linear Search
O(log n): Binary Search
O(n log n): Quick Sort / Heap Sort / Merge Sort

ii) Identify the best performing algorithm. **1**

Mark scheme

O(log n) (1 mark)

c) An algorithm has two nested loops. Each loop completes after **N** counts. How many counts will it take for both loops to complete? What is the *Big O notation* for this algorithm? **2**

Mark scheme

N * N or N^2 counts (1 mark)
O(N^2) or O(n^2) (1 mark)

Total 10 Marks

Marks

Question 9

a) The following program breaks down a sentence into its individual words which are then stored in an array of strings. This is called *parsing* the sentence.

```
01. data words as array of (2) string
02. data sentence as string
03. data n as whole number
04. data p as whole number

05. sentence = "Hello Ali"
06. n = 0             //index into string variable sentence
07. p = 0             //index into array variable words

08. Loop until n is equal to sizeof(sentence)
09.    if sentence[n] not equal to " " then
10.       words[p] = words[p] + sentence[n]
11.    else
12.       p = p + 1  //increment index into words array
13.    end if
14.    n = n + 1     //increment index into sentence string
15. Next loop
```

i) Explain what line 10 is doing. **1**

Mark scheme

Individual letters of the string variable sentence are appended to the array's string elements. (1 mark)
****** Accept alternative correct explanation.***

Marks

ii) Copy the following table and fill in the values of the data variables at line 10. **9**
The values of variable **n** are already filled in.

n	p	sentence[n]	words[p]
0			
1			
2			
3			
4			
5			
6			
7			
8			

Mark scheme

n	*p*	*sentence[n]*	*words[p]*
0	***0***	***H***	***H***
1	***0***	***e***	***He***
2	***0***	***l***	***Hel***
3	***0***	***l***	***Hell***
4	***0***	***o***	***Hello***
5	***1***	***space***	
6	***1***	***A***	***A***
7	***1***	***l***	***Al***
8	***1***	***i***	***Ali***

1 mark for each correct row (maximum 9 marks).

At row n=5, the words[p] entry should be left as blank or with some other meaningful entry. However, if it is filled in with an 'A' and the sequence of subsequent entries are correct, i.e. 'Al' followed by 'Ali', then allow but deduct 1 mark.

Total 10 Marks

Marks

Question 10

Given the following program:

```
01 Class Student
02    data name as String
03    data gender as Character
04    data age as Whole number

05    Function Student(needs n as String, g as Character,
06                     a as Whole number)
07       name = n
08       gender = g
09       age = a
10    End function

11    Function setName(needs newName as String)
12       name = newName
13    End function

14    Function setAge(needs newAge as String)
15       age = newAge
16    End function
17 End class
```

i) Identify the *constructor method.* **1**

Mark scheme

Student (1 mark)

ii) Identify the *accessor methods.* **2**

Mark scheme

setName, setAge (1 mark for each, maximum 2 marks)

b) Add a method to Student class that queries the student's age. **3**

Mark scheme

```
Function getAge() returns Whole number
   Return age
End function
```

1 mark for the correct function structure (Function/End function).
1 mark for the correct return type of the function.
1 mark for the correct return statement.
****** Accept alternative function name.***

Marks

c) Write the code necessary to do the following actions: **4**

- Create the new object **myStudent** of type **Student** using suitable parameters.
- Print the age of **myStudent.**

Mark scheme

data myStudent as Student
myStudent = new Student("Liz", 'F', 17)
output myStudent.getAge
****** Accept any suitable parameters***
****** Accept output myStudent→age***

1 mark for correct declaration of myStudent.
1 mark for the use of the 'new' keyword.
1 mark for the correct use of the constructor parameter.
1 mark for the correct printing of the age of myStudent.

Total 10 Marks

End of paper

Marks

Learning Outcomes matrix

Question	Learning Outcomes assessed	Marker can differentiate between varying levels of achievement
1	LO1	Yes
2	LO6	Yes
3	LO2	Yes
4	LO3	Yes
5	LO3	Yes
6	LO4, LO5	Yes
7	LO3	Yes
8	LO5	Yes
9	LO4, LO5	Yes
10	LO7	Yes

Grade descriptors

Learning Outcome	Fail	Referral	Pass	Merit	Distinction
Identify and explain the key stages of software development lifecycles	Can basically identify, adapt and use appropriate skills, methods and procedures to reach basic solutions.	In a limited way, can identify, adapt and use appropriate skills, methods and procedures to reach limited solutions.	Can adequately Identify, adapt and use appropriate skills, methods and procedures to reach appropriate solutions.	Can soundly identify, adapt and use appropriate skills, methods and procedures to reach supported and appropriate solutions.	Can coherently identify, adapt and use appropriate skills, methods and procedures to reach well supported and highly appropriate solutions.
Express, design and evaluate algorithms	Demonstrates basic ability to review the effectiveness and appropriateness of actions, methods and results	Demonstrates limited ability to review the effectiveness and appropriateness of actions, methods and results	Demonstrates adequate ability to review the effectiveness and appropriateness of actions, methods and results	Demonstrates sound ability to review the effectiveness and appropriateness of actions, methods and results	Demonstrates comprehensive ability to review the effectiveness and appropriateness of actions, methods and results
Identify and use programming language constructs	Can basically identify, adapt and use appropriate skills, methods and procedures to reach basic solutions.	In a limited way, can identify, adapt and use appropriate skills, methods and procedures to reach limited solutions.	Can adequately Identify, adapt and use appropriate skills, methods and procedures to reach appropriate solutions.	Can soundly identify, adapt and use appropriate skills, methods and procedures to reach supported and appropriate solutions.	Can coherently identify, adapt and use appropriate skills, methods and procedures to reach well supported and highly appropriate solutions.
Identify and use common	Can basically identify, adapt and use appropriate	In a limited way, can identify, adapt and use appropriate	Can adequately Identify, adapt and use appropriate	Can soundly identify, adapt and use appropriate	Can coherently identify, adapt and use appropriate

					Marks
data structures	skills, methods and procedures to reach basic solutions.	skills, methods and procedures to reach limited solutions.	skills, methods and procedures to reach appropriate solutions.	skills, methods and procedures to reach supported and appropriate solutions.	skills, methods and procedures to reach well supported and highly appropriate solutions.
Explain and use common algorithms	Demonstrates basic ability to review the effectiveness and appropriateness of actions, methods and results	Demonstrates limited ability to review the effectiveness and appropriateness of actions, methods and results	Demonstrates adequate ability to review the effectiveness and appropriateness of actions, methods and results	Demonstrates sound ability to review the effectiveness and appropriateness of actions, methods and results	Demonstrates comprehensive ability to review the effectiveness and appropriateness of actions, methods and results
Explain and use test strategies	Demonstrates basic ability to review the effectiveness and appropriateness of actions, methods and results	Demonstrates limited ability to review the effectiveness and appropriateness of actions, methods and results	Demonstrates adequate ability to review the effectiveness and appropriateness of actions, methods and results	Demonstrates sound ability to review the effectiveness and appropriateness of actions, methods and results	Demonstrates comprehensive ability to review the effectiveness and appropriateness of actions, methods and results
Explain how software is modularised	Demonstrates basic ability to review the effectiveness and appropriateness of actions, methods and results	Demonstrates limited ability to review the effectiveness and appropriateness of actions, methods and results	Demonstrates adequate ability to review the effectiveness and appropriateness of actions, methods and results	Demonstrates sound ability to review the effectiveness and appropriateness of actions, methods and results	Demonstrates comprehensive ability to review the effectiveness and appropriateness of actions, methods and results