Version 1.0: 0107

ASSESSMENT and QUALIFICATIONS ALLIANCE

General Certificate of Education

Computing 6510

CPT4 Processing and Programming Techniques

Mark Scheme

2007 examination - January series

Mark schemes are prepared by the Principal Examiner and considered, together with the relevant questions, by a panel of subject teachers. This mark scheme includes any amendments made at the standardisation meeting attended by all examiners and is the scheme which was used by them in this examination. The standardisation meeting ensures that the mark scheme covers the candidates' responses to questions and that every examiner understands and applies it in the same correct way. As preparation for the standardisation meeting each examiner analyses a number of candidates' scripts: alternative answers not already covered by the mark scheme are discussed at the meeting and legislated for. If, after this meeting, examiners encounter unusual answers which have not been discussed at the meeting they are required to refer these to the Principal Examiner.

It must be stressed that a mark scheme is a working document, in many cases further developed and expanded on the basis of candidates' reactions to a particular paper. Assumptions about future mark schemes on the basis of one year's document should be avoided; whilst the guiding principles of assessment remain constant, details will change, depending on the content of a particular examination paper.

Instructions to examiners

The following forms of notation should be used on candidates' scripts:

- Ticks - To indicate what is accepted as correct or creditworthy, placed in the body of the answer, and on diagrams;
- Underscoring – To identify errors/irrelevance in written answers;
- Crosses – to indicate a wrong answer;
- Brief comments – placed at suitable points in the body of the text to amplify the marking;
- BOD – means benefit of the doubt and is used where the candidate's answer has been given a mark on the balance of probabilities that the candidate's answer has met the requirements of the mark scheme even though it could be interpreted differently;
- NE – means not enough and is applied to an answer that falls short of what is required;
- O/S – means outside the mark scheme. The candidate's answer is creditworthy but the answer does not match any of the answers on the mark scheme for the particular question. Nevertheless a mark is awarded;
- C/F – means carried forward. This arises when a candidate offers an answer which is not creditworthy in one question but is creditworthy in a later question. The mark is carried forward to the question which is creditworthy;
- C/B – means carried back. This is similar to a carry forward but the mark is carried back to an earlier question.
- T/O – means talked out. The candidate's answer is contradictory.
- ^ - means missing term or symbol.
- F/T – means followed through. If a candidate made a mistake in the earlier part of an answer, mar the answer using the correct method on their answer from the earlier part.

The following notation is used in the mark scheme

- ; - means a single mark;
- **A** – means acceptable creditworthy answer;
- **R** – means reject answer as not creditworthy;
- **I** – means ignore.
- / - means alternative word or sub-phrase;
- // - means alternative answer

<u>General Rules for Marking</u>
Ignore Abbreviations
Ignore Brand Names

1. (a) 40E **1**

(b) 1038 **1**

(c) 64.875 1 mark for 64, 1mark for .875 **A** 7/8 **2**

(d) (i) 0.125//$^1/_8$;;;
If incorrect part marks as follows
mantissa = 0.5 or ½ 1 mark
exponent = -2 1 mark
times 2^{exponent} 1 mark **3**

(ii) leftmost 2 digits/bits are different//
a significant bit is stored after the binary point//
bit after point different from bit before point;
A the first bit after the sign bit is a '1';
A The second bit is a '1';
A *an answer that <u>clearly</u> implies a '1' follows the '0'* **1**

(iii) 127;;//1111111;;//0.1111111; x 2^7/2^{111}; **2**

Total 10

2.

Component	Name
1	Program Counter
2	Memory Address Register; **A** MAR
3	Address Bus;
4	Data Bus;
5	Memory Data Register/ Memory Buffer Register; **A** MDR/MBR
6	Current Instruction Register; A Instruction Register/IR/CIR

5

Total 5

3. (a) hall (kingston);
resident (richard,kingston);
studies (richard,computing);
I order **3**

(b) jayesh; tanya;
I Order
Penalise each additional entry **2**

(c) resident(Name,Hall); AND; studies(Name,Subject);//
resident(Name,Hall); AND; studies(Name,Subject); AND hall(Hall)
I Order **3**

Penalise wrong case once in each section

Total 8

4 (a) (i) Processing continues from beginning to end without user intervention//
Program run in background//
Processing delayed until all data has been entered; **1**

(ii) User and computer in two way communication//
Processing carried out as user enters the data; **1**

(b) jobID; priority; user name; job delimiters; job completion time;
approximate execution time; max length of time job can run for;
start time of job; main memory required; devices/hardware required;
compiler/assembler/software required; name of file/program to be executed;
what to do on successful completion of job; what to do in case of error;
data file required; output destination; **max 3**

(c) (i) Interactive; **1**

(ii) User requires immediate response//
Interactive jobs have shorter burst times; **1**

(iii) Last burst time; Priority given to the job by the user;
Proximity of deadline for a batch job;
Proximity of end of job;
Time waiting in inactive list;
Resources required;
Estimated running time;
I/O bound or processor bound; **max 2**

Total 9

5 (a) need to access/address registers/ exact memory addresses/ hardware directly;
fast speed of operation required; code needs to take up little memory;
A minimise the size of the program code;
A no compiler/interpreter exists yet for the machine// no other translator exists;
R manipulate bits **2**

5 (b)

Label	Opcode	Operand	Comment
X	DEFB		Declare variable X
COUNT	DEFB		Declare variable COUNT
	LD	#00	Store 0
	ST	X	in X;
	LD	#01	Initialise
	ST	COUNT	COUNT to 1;
LOOP;	LD	COUNT	If Count
	CMP	#05	Is greater than 5;
	JG	NEXT	Exit the for loop;
	LD	X	Load X
	ADD	COUNT	Add COUNT
	ST	X	And store result in X;
	LD	COUNT	Add
	ADD	#01	1
	ST	COUNT	To COUNT;
	JP	LOOP	Go back and test for end of loop;
NEXT;			

Alternative

Label	Opcode	Operand	Comment
X	DEFB	#00	Declare variable X and initialise to 0;
COUNT	DEFB	#01	Declare variable COUNT and initialise to 1;
LOOP;	LD	COUNT	If COUNT
	CMP	#05	Is greater than 5;
	JG	NEXT	Exit the for loop;
	LD	X	Load X
	ADD	COUNT	Add COUNT
	ST	X	And store result in X;
	LD	COUNT	Add
	ADD	#01	1
	ST	COUNT	To COUNT;
	JP	LOOP	Go back and test for end of loop;
NEXT;			

Alternative

Label	Opcode	Operand	Comment
X	DEFB	#00	Declare variable X and initialise to 0;
COUNT	DEFB	#00	Declare variable COUNT and initialise to 0;
LOOP;	LD	COUNT	Load COUNT
	ADD	#01	Add1
	ST	COUNT	And store result in COUNT;
	CMP	#05	If COUNT is greater than 5;
	JG	NEXT	Exit the for loop;
	ADD	X	Add X to COUNT
	ST	X	And store result in X;
	JP	LOOP	Go back and test for end of loop;
NEXT;			

A an answer that performs the test at the end of the loop **8**

Total 10

6 (a)

```
        Temp ← Front;
        Front ← Temp.Next//Front ← Temp^.Next;
        Dispose (Temp);        A Free(Temp)
Alternative
        Temp ← Front.Next// Temp ← Front^.Next;
        Dispose (Front);       A Free(Temp)
        Front ← Temp;
```

3

(b) AddItem//Add; **1**

(c) (i) Full/FullQueue; **1**

(ii) No memory used for pointers;
I Faster
R Easier to program **1**

(iii) Size is limited by array size;
memory wasted when not full; **2**

Total 8

7 (a)

Number	Lower	Upper	Current
12	1	9	
		5	5
	3		3
	4	4	4

Value returned	4

1 mark for 1st row (12, 1, 9)
2 marks for second row (1 mark for each 5)
2 marks for 3rd row (3 and 3)
2 marks for 4th row (1 mark for Lower = 4, 1mark for upper = 4)
1 mark for correct return value **8**

(b) Find the position of 12/ a number in the array// search for 12/ a number in the array; **1**

Total 9

8	(a)	mouse click// mouse movement// keyboard operation// any interrupt;	**1**
	(b)	event-driven programs service an event and wait for another; non event-driven programs run to completion/ are sequential;	**2**
	(c)	contains its own data/fields/variables/properties; contains its own operations/methods/functions/procedures/behaviours/code; responds to messages; **A** Based on a Class definition	**max 2**
	(d)	frame/form/window/button/check box/radio button/menu/text box; **A** any sensible widget **R** Plurals	**1**
		Total	**6**