This mark scheme is published as an aid to teachers and candidates, to indicate the requirements of the examination. It shows the basis on which Examiners were instructed to award marks. It does not indicate the details of the discussions that took place at an Examiners’ meeting before marking began, which would have considered the acceptability of alternative answers.

Mark schemes should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge will not enter into discussions about these mark schemes.

Cambridge is publishing the mark schemes for the May/June 2015 series for most Cambridge IGCSE®, Cambridge International A and AS Level components and some Cambridge O Level components.

© IGCSE is the registered trademark of Cambridge International Examinations.
Mark as follows:
1 mark for both states correct
1 mark for each further label [5]

2 (a) capital_city(santiago).
city_in_country(santiago, chile).
country_in_continent(chile, south_america).
city_visited(santiago).

accept in any order [4]

(b) ThisCity =
manchester
london [2]

(c) countries_visited(ThisCountry)
IF
city_visited(ThisCity) 1
AND 1
city_in_country(ThisCity, ThisCountry) 2 [4]
### (a)

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Y</th>
<th>Y</th>
<th>Y</th>
<th>Y</th>
<th>N</th>
<th>N</th>
<th>N</th>
<th>N</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>goods totalling more than $20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>goods totalling more than $100</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>have discount card</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actions</th>
<th>1 mark</th>
<th>1 mark</th>
<th>1 mark</th>
<th>1 mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>No discount</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>5% discount</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10% discount</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[4 marks]

### (b)

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Y</th>
<th>Y</th>
<th>Y</th>
<th>Y</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>goods totalling more than $20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>goods totalling more than $100</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>have discount card</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actions</th>
<th>1 mark per column</th>
</tr>
</thead>
<tbody>
<tr>
<td>No discount</td>
<td>X</td>
</tr>
<tr>
<td>5% discount</td>
<td>X</td>
</tr>
<tr>
<td>10% discount</td>
<td>X</td>
</tr>
</tbody>
</table>

[5 marks]
(c) Example Pascal

FUNCTION Discount(GoodsTotal: INTEGER; HasDiscountCard: BOOLEAN) : INTEGER;

BEGIN
(1) IF GoodsTotal > 20
(1) THEN
(2) IF GoodsTotal > 100
(2) THEN
(3) IF HasDiscountCard = TRUE
(3) THEN
(3) Discount := 10
(3) ELSE
(3) Discount := 5
(3) ELSE
(4) IF HasDiscountCard = TRUE
(4) THEN
(4) Discount := 5
(4) ELSE
(4) Discount := 0
(1) ELSE
(1) Discount := 0;
END;

Example Python

def Discount(GoodsTotal, HasDiscountCard) :

(1) if GoodsTotal > 20:
(2)   if GoodsTotal > 100:
(3)     if HasDiscountCard == True:
(3)       return 10
(3)     else:
(3)       return 5
(2)   else:
(4)     if HasDiscountCard == True:
(4)       return 5
(4)     else:
(4)       return 0
(1) else:
(1)   return 0
(b) Example Pascal

Type
Employee = CLASS
  PUBLIC
  procedure SetEmployeeName
  Procedure SetEmployeeID
  Procedure CalculatePay
  PRIVATE
  EmployeeName : STRING
  EmployeeID : STRING
  AmountPaidThisMonth : Currency
END;

Mark as follows:
Class header (1 mark)
PUBLIC and PRIVATE used correctly (1 mark)
EmployeeName + EmployeeID (1 mark)
AmountPaidThisMonth (1 mark)
Methods x 3 (1 mark)

Example VB

Class Employee
  Private EmployeeName As String
  Private EmployeeID As String
  Private AmountPaidThisMonth As Decimal
Public Sub SetEmployeeName()
End Sub
Public Sub SetEmployeeID()
End Sub
Public Sub CalculatePay()
End Sub

Example Python

Class Employee():
  def __init__(self):
    self.__EmployeeName = ""
    self.__EmployeeID = ""
    self.__AmountPaidThisMonth = 0
  def SetEmployeeName(self, Name):
    self.__EmployeeName = Name
  def SetEmployeeID(self, ID):
    self. EmployeeID = ID
  def SetAmountPaidThisMonth(self, Paid):
    self.__AmountPaidThisMonth = Paid

[max 5]
(c) (i) HoursWorked 1
HourlyPayRate 1
SetHoursWorked 1
CalculatePay : Override 1 + 1
SetPayRate 1
\[\text{max 4}\]

(ii) AnnualSalary 1
SetSalary 1
CalculatePay : Override 1
\[\text{max 2}\]

(d) Polymorphism [1]

5 (a) (i) FOR ThisPointer \(\leftarrow 2 \text{ TO } 10\)
\[\text{1 mark for each gap filled correctly}\]
\[\text{7}\]

(ii) The outer loop (FOR loop) is executed 9 times (1 mark)
it is not dependant on the dataset (1 mark)

The Inner loop (WHILE loop) is not entered (1 mark)
as the condition is already false at the first encounter (1 mark) [max 3]

(b) (i) outer loop is executed 9 times (1 mark)
inner loop is executed 9 times (for each iteration of the outer loop) (1 mark)
not dependant on the dataset (1 mark) [max 2]
(ii) \( \text{NumberOfItems} \leftarrow 10 \)

\[
\text{REPEAT} \\
\quad \text{NoMoreSwaps} \leftarrow \text{TRUE} \\
\quad \text{FOR} \text{ Pointer} \leftarrow 1 \text{ TO } \text{NumberOfItems} - 1 \text{ \text{\textbf{\textit{UNTIL}} NoMoreSwaps = TRUE}} \\
\quad \text{IF NameList[Pointer]} > \text{NameList[Pointer + 1]} \\
\quad \quad \text{THEN} \\
\quad \quad \quad \text{NoMoreSwaps} \leftarrow \text{FALSE} \\
\quad \quad \quad \text{Temp} \leftarrow \text{NameList[Pointer]} \\
\quad \quad \quad \text{NameList[Pointer]} \leftarrow \text{NameList[Pointer + 1]} \\
\quad \quad \quad \text{NameList[Pointer + 1]} \leftarrow \text{Temp} \\
\quad \quad \text{ENDIF} \\
\text{ENDFOR} \\
\text{NumberOfItems} \leftarrow \text{NumberOfItems} - 1 \\
\text{UNTIL NoMoreSwaps = TRUE}
\]

Mark as follows:
- change outer loop to a **REPEAT/WHILE** loop \( (1 \text{ mark}) \)
- **FOR** loop has variable used for final value \( \text{ (1 mark)} \)
- Initialise Boolean variable to **TRUE** \( (1 \text{ mark}) \)
- set Boolean variable to **FALSE** in correct place \( (1 \text{ mark}) \)
- number of items to consider on each pass decrements \( (1 \text{ mark}) \)
- Correct stopping condition for **REPEAT** loop \( (1 \text{ mark}) \) \[\text{max 5}\]

6 (a)

\[\text{\begin{array}{c}
\text{Ben} \\
\text{Ahmed} \\
\text{Jatinder}
\end{array}}\]

1 mark for **Head and Tail** pointers
1 mark for 3 correct items – linked as shown
1 mark for correct order with null pointer in last nod \[\text{[3]}\]
(b) (i) 

Queue

<table>
<thead>
<tr>
<th>HeadPointer</th>
<th>Name</th>
<th>Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[1]</td>
<td>2</td>
</tr>
<tr>
<td>[2]</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>[3]</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>[4]</td>
<td>5</td>
</tr>
<tr>
<td>[5]</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>[6]</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>[7]</td>
<td>8</td>
</tr>
<tr>
<td>[8]</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>[9]</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>[10]</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Mark as follows:

HeadPointer = 0 & TailPointer = 0
FreePointer assigned a value
Pointers[1] to [9] links the nodes together
Pointer[10] = 'Null'

(ii) PROCEDURE RemoveName()
// Report error if Queue is empty
IF HeadPointer = 0
THEN
    Error
ELSE
    OUTPUT Queue[HeadPointer].Name
    // current node is head of queue
    CurrentPointer ← HeadPointer
    // update head pointer
    HeadPointer ← Queue[CurrentPointer].Pointer
    // if only one element in queue, then update tail pointer
    IF HeadPointer = 0
    THEN
        TailPointer ← 0
    ENDIF
    // link released node to free list
    Queue[CurrentPointer].Pointer ← FreePointer
    FreePointer ← CurrentPointer
ENDIF
ENDPROCEDURE