1(a)

Answers include:

1 They are tried and tested // Free from errors
2 They are readily available // Speed up development time
3 They will be updated automatically when improvements are made // No maintenance of routine is needed (as automatically updated)
4 The algorithm will be compliant with the international standard

One mark per point

Max 3

1(b)(i)

1 When a part of the algorithm performs a specific task
2 Part is repeated // performed in several places
3 Testing / debugging / maintenance is easier

One mark for each point

Max 2

1(b)(ii)

Term	Description
`Pass2`	the name of the function
`Count`	The parameter / value passed to the function
`BOOLEAN`	The (data) **type** returned (by the function)

One mark per description (excluding first row)

1(c)

Expression	Evaluates to
`LENGTH(NUM_TO_STR(Multiplier))`	**3**
`MONTH(DoB) > 4`	**TRUE**
`15 + STR_TO_NUM(MID(AddressLine[1], 2, 1))`	**20**

One mark per row

2(a)

1 mark each to max
- Record structure or class with constructor `Queue` (and end where appropriate) …
- … containing a 1D array of (100) integers `QueueArray`
- … containing `HeadPointer` and `TailPointer` as integers

e.g.
Python
```
class Queue:
    def __init__(self):
        self.QueueArray = []
        HeadPointer = 0 #integer
        TailPointer = 0 #integer
        for x in range(0, 100):
            self.QueueArray.append(-1)
```

VB.NET
```
Structure Queue
    Dim QueueArray() As Integer
    Dim HeadPointer As Integer
    Dim TailPointer As Integer
End Structure
```

Java
```
class queue{
     private static Integer[] QueueArray = new Integer[100];
     private static Integer HeadPointer;
     private static Integer TailPointer;

     public queue(){
     }
}
```

2(b)

1 mark each
- New `Queue` record/object created/instance of class
- Queue **field/attribute** head pointer initialised to `-1`, tail pointer to `0`
- All 100 array **field/attribute** elements initialised with `-1`

e.g.
Python
```
class Queue:
    def __init__(self):
        self.QueueArray = []
        for x in range(0, 100):
            self.QueueArray.append(-1)

        self.HeadPointer = -1
        self.TailPointer = 0
TheQueue= Queue()
```

VB.NET
```
 Dim TheQueue As New Queue
        TheQueue.HeadPointer = -1
        TheQueue.TailPointer = 0
        ReDim TheQueue.QueueArray(100)
        For x = 0 To 99
            TheQueue.QueueArray(x) = -1
        Next
```

Java
```
class queue{
     private static Integer[] QueueArray = new Integer[100];
     private static Integer HeadPointer;
     private static Integer TailPointer;
```

2(b)

```
      public queue(){
            HeadPointer = -1;
            TailPointer = 0;
            for(Integer x = 0; x < 100; x++){
                  QueueArray[x] = -1;
            }
      }
}
public static void main(String args[]){
      queue TheQueue = new queue();
}
```

2(c)

1 mark for each completed statement to max 3
1 mark for correct values returned in correct places
1 mark for function header taking (at least) one parameter and the rest of function correct and using the record/class data structure accurately.

Pseudocode

```
FUNCTION Enqueue(BYREF AQueue : Queue, BYVAL TheData : INTEGER)
                  RETURNS INTEGER
   IF AQueue.HeadPointer = -1 THEN
      AQueue.QueueArray[AQueue.TailPointer] ← TheData
      AQueue.HeadPointer ← 0
      AQueue.TailPointer ← AQueue.TailPointer + 1
      RETURN 1
   ELSE
      IF AQueue.TailPointer > 99 THEN
        RETURN -1
      ELSE
        AQueue.QueueArray[AQueue.TailPointer] ← TheData
        AQueue.TailPointer ← AQueue.TailPointer + 1
        RETURN 1
      ENDIF
   ENDIF
ENDFUNCTION
```

2(c)

e.g.

Python

```
def Enqueue(AQueue, TheData):

    if AQueue.HeadPointer == -1:
        AQueue.HeadPointer = 0
        AQueue.QueueArray[AQueue.HeadPointer] = TheData
        AQueue.TailPointer +=1
        return AQueue, 1
    elif AQueue.TailPointer > 99:
        return AQueue, -1
    else:
        AQueue.QueueArray[AQueue.TailPointer] = TheData
        AQueue.TailPointer = AQueue.TailPointer + 1
        return AQueue, 1
```

VB.NET

```
Function Enqueue(ByRef AQueue As Queue, ByVal TheData As Integer)
     If AQueue.HeadPointer = -1 Then
         AQueue.QueueArray(AQueue.TailPointer) = TheData
         AQueue.HeadPointer = 0
          AQueue.TailPointer += 1
         Return 1
     ElseIf AQueue.TailPointer > 99 Then
         Return -1
     Else
         AQueue.QueueArray(AQueue.TailPointer) = TheData
         AQueue.TailPointer += 1
         Return 1
     End If

End Function
```

2(c)

Java

```
public static Integer Enqueue(Integer TheData){
      if(GetHeadPointer() == -1){
            SetData(TheData);
            SetHeadPointer(0);
            SetTailPointer(GetTailPointer() + 1);
            return 1;
      }else if(GetTailPointer() > 99){
            return -1;
      }else{
            SetData(TheData);
            SetTailPointer(GetTailPointer() + 1);
            return 1;
      }
}
```

2(d)

1 mark each to max 3

- Function header (and end) iterating through each element in the queue
- Starting at `HeadPointer` and incrementing until `TailPointer - 1` …
- … concatenating and returning all integer values with a space between

e.g.
Python

```
def ReturnAllData(TheQueue):
    Temp = ""

    for X in range(TheQueue.HeadPointer, TheQueue.TailPointer):
        Temp = Temp + str(TheQueue.QueueArray[X]) + " "
    return Temp
```

2(d)	VB.NET
	`Function ReturnAllData(AQueue As Queue)`
	`  Dim Temp As String = ""`
	`  For X = AQueue.HeadPointer To AQueue.TailPointer - 1`
	`  Temp = Temp & AQueue.QueueArray(X).ToString() & " "`
	`  Next X`
	`  Return Temp`
	`End Function`
	
	Java
	`public static String ReturnAllData(){`
	`  String Temp = "";`
	`  Integer Counter = 0;`
	`  for(int X = HeadPointer; X < TailPointer; X++){`
	`  Temp = Temp + Integer.toString(QueueArray[X]) + " ";`
	`  }`
	`  return Temp;`
	`}`

2(e)(i)	1 mark each • Taking only **10** inputs in loop/one at a time • Calling `Enqueue()` with each input (min) **and** storing/using return value … • … only calling `Enqueue()` **once** when each input is an integer > = 0. Do **not** award if this validation stops 10 valid inputs being enqueued. • Outputting message if each item is inserted **and** outputting a message if queue is full • Calling `ReturnAllData()` and outputting return value at the end

2(e)(i)

e.g.

Python

```
for x in range(0, 10):
    Continue = True
    while(Continue == True):
        DataInput = int(input("Enter an integer that is 0 or more"))
        if DataInput > -1:
            Continue = False

    TheQueue, ReturnValue = Enqueue(TheQueue, DataInput)

    if(ReturnValue == -1):
        print("Queue full")
    else:
        print("Item inserted")

print(ReturnAllData(TheQueue))
```

VB.NET

```
Dim ContinueLoop As Boolean
Dim DataInput As Integer
Dim ReturnValue As Integer
For x = 0 To 9
    ContinueLoop = True
    While ContinueLoop = True
        Console.WriteLine("Enter an integer that is 0 or more")
        DataInput = Console.ReadLine
        If DataInput > -1 Then
            ContinueLoop = False
```

2(e)(i)

```
        End If
    End While
    ReturnValue = Enqueue(TheQueue, DataInput)
    If ReturnValue = 2 Then
        Console.WriteLine("Queue full")
    Else
        Console.WriteLine("Item inserted")
    End If
Next

Console.WriteLine(ReturnAllData(TheQueue))
```

Java

```
Boolean Continue = true;
Integer DataInput = -1;
Scanner scanner = new Scanner(System.in);
Integer ReturnValue;

for(Integer x = 0; x < 10; x++){
      Continue = true;
      while(Continue == true){
            System.out.println("Enter an integer that is 0 or more");
            DataInput = Integer.parseInt(scanner.nextLine());
            if(DataInput > -1){
                  Continue = false;
            }
      }
      ReturnValue = Enqueue(DataInput);
      if(ReturnValue == -1){
            System.out.println("Queue full");
      }else{
            System.out.println("Item inserted");
      }
}
System.out.println(ReturnAllData());
```

2(e)(ii)	1 mark for each • All values input and 10 messages 'Inserted' (i.e. –1 is not inserted) • Screenshot show 10 9 8 7 6 5 4 3 2 1 on one line with a space between each number e.g. Enter an integer that is 0 or more10 Item inserted Enter an integer that is 0 or more9 Item inserted Enter an integer that is 0 or more-1 Enter an integer that is 0 or more8 Item inserted Enter an integer that is 0 or more7 Item inserted Enter an integer that is 0 or more6 Item inserted Enter an integer that is 0 or more5 Item inserted Enter an integer that is 0 or more4 Item inserted Enter an integer that is 0 or more3 Item inserted Enter an integer that is 0 or more2 Item inserted Enter an integer that is 0 or more1 Item inserted 10 9 8 7 6 5 4 3 2 1

2(f)	1 mark each
	• Function `Dequeue()` (head and close), returning a value in all cases
	• Checking if empty **and** returning –1
	• Returning item at `HeadPointer` **without deleting**/changing it
	• Incrementing `HeadPointer`
	
	Example program code:
	Python
	`def Dequeue(AQueue):`
	`    if AQueue.HeadPointer = 100 or AQueue.HeadPointer == -1 or AQueue.HeadPointer == AQueue.TailPointer:`
	`        return AQueue, -1`
	`    else:`
	`        Temp = AQueue.QueueArray[AQueue.HeadPointer]`
	`        AQueue.HeadPointer = AQueue.HeadPointer + 1`
	`        return AQueue, Temp`

2(f)

VB.NET

```
  Function Dequeue(ByRef AQueue As Queue)
        If AQueue.HeadPointer = 100 or AQueue.HeadPointer = -1 or AQueue.HeadPointer =
AQueue.TailPointer Then
            Return -1
        Else
            Dim Temp As Integer = AQueue.QueueArray(AQueue.HeadPointer)
            AQueue.HeadPointer += 1
            Return Temp
        End If
  End Function
```

Java

```
public static Integer Dequeue(){
     if(GetHeadPointer() = 100 || GetTailpointer() == -1 || GetHeadPointer() ==
GetTailpointer()){
           return -1;
     }else{
           Integer Temp = GetData(GetHeadPointer());
           SetHeadPointer(GetHeadPointer() + 1);
           return Temp;
     }
}
```

2(g)(i)	1 mark each
	• Calls `Dequeue()` **twice and** stores/uses return value
	• Outputs "Queue empty" when each return values is –1 and outputs return value otherwise
	• Calls `ReturnAllData()` at the end
	
	e.g.
	Python
	`TheQueue, ReturnValue = Dequeue(TheQueue)`
	`if ReturnValue == -1:`
	`  print("Queue empty")`
	`else:`
	`  print(ReturnValue, " is returned")`
	`TheQueue, ReturnValue = Dequeue(TheQueue)`
	`if ReturnValue == -1:`
	`  print("Queue empty")`
	`else:`
	`  print(ReturnValue, " is returned")`
	`print(ReturnAllData(TheQueue))`
	
	VB.NET
	`ReturnValue = Dequeue(TheQueue)`
	`If ReturnValue = -1 Then`
	`  Console.WriteLine("Queue empty")`
	`Else`
	`  Console.WriteLine(ReturnValue, " is returned")`
	`End If`

2(g)(i)

```
ReturnValue = Dequeue(TheQueue)
If ReturnValue = -1 Then
    Console.WriteLine("Queue empty")
Else
    Console.WriteLine(ReturnValue, " is returned")
End If
Console.WriteLine(ReturnAllData(TheQueue))
```

Java

```
ReturnValue = Dequeue();
if(ReturnValue == -1){
      System.out.println("Queue empty");
}else{
      System.out.println(ReturnValue + " is returned");
}

ReturnValue = Dequeue();
if(ReturnValue == -1){
      System.out.println("Queue empty");
}else{
      System.out.println(ReturnValue + " is returned");
}
ReturnAllData(TheQueue);
```

2(g)(ii)

1 mark each

- Screenshot shows the input of 10 9 8 7 6 5 4 3 2 1

Output for 10 returned
Output for 9 is returned

e.g.

```
Enter an integer that is 0 or more10
Item inserted
Enter an integer that is 0 or more9
Item inserted
Enter an integer that is 0 or more-1
Enter an integer that is 0 or more8
Item inserted
Enter an integer that is 0 or more7
Item inserted
Enter an integer that is 0 or more6
Item inserted
Enter an integer that is 0 or more5
Item inserted
Enter an integer that is 0 or more4
Item inserted
Enter an integer that is 0 or more3
Item inserted
Enter an integer that is 0 or more2
Item inserted
Enter an integer that is 0 or more1
Item inserted
10 9 8 7 6 5 4 3 2 1
10  is returned
9  is returned
8 7 6 5 4 3 2 1
```