8 An exam paper has a maximum of 75 marks. One of five pass grades (A to E) is assigned, depending on the mark obtained. The lowest mark for a given grade is known as the grade boundary. For example, if the grade boundary for an A grade is 65 marks, then any candidate who achieves a mark of 65 or above will be awarded an A. A grade of U is awarded for marks below the E grade boundary.

The five grade boundaries are stored in a global 1D array `GradeBoundary` of type integer.

For example:

Element	**Value**	**Comment**
`GradeBoundary[1]`	65	The minimum mark for an A grade.
`GradeBoundary[2]`	57	The minimum mark for a B grade.
`GradeBoundary[3]`	43	The minimum mark for a C grade.
`GradeBoundary[4]`	35	The minimum mark for a D grade.
`GradeBoundary[5]`	27	The minimum mark for an E grade.

A global 2D array `Result` of type integer contains candidate marks for the exam. Each row relates to one candidate. Column 1 contains the candidate mark and column 2 contains the unique candidate ID.

For example, for the fourth and fifth candidates:

Element	**Mark**
`Result[4, 1]`	`56`
`Result[5, 1]`	`54`

Element	**ID**
`Result[4, 2]`	`1074832`
`Result[5, 2]`	`2573839`

There are more rows in the array than candidates who sit the exam. Any unused rows will be at the end of the array.

Candidate papers that are given a mark within two marks of any grade boundary must be checked.

For example, given the values in the example grade boundaries above, any paper that was awarded between 41 and 45 marks (inclusive) would need to be checked.

A program is being written to identify papers that need to be checked.

The programmer has defined the first program module as follows:

Module	**Description**
`CheckMark()`	• called with a parameter of type integer representing a candidate mark • returns `TRUE` if the mark is within 2 of any of the five grade boundaries, otherwise returns `FALSE`

(a) Write pseudocode for module `CheckMark()`.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [6]

(b) A second module is defined:

Module	**Description**
`CheckAll()`	• called with a parameter of type integer representing the number of candidate marks in the `Result` array • uses `CheckMark()` to check each candidate mark • for each paper that needs to be checked, write the corresponding candidate ID on a separate line in a new file named `GRList.txt` • outputs a message with a count of how many papers need to be checked

Write pseudocode for module `CheckAll()`.

`CheckMark()` must be used to check each individual mark.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [8]

(c) The requirement changes. Instead of a new file, the module described in part **(b)** needs to add the corresponding candidate ID for each paper that needs to be checked to an **existing** file.

Explain the change that will need to be made to `CheckAll()`.

....

.... [1]