1 Suppose we are comparing implementations of insertion sort and merge sort on the same machine. For inputs of size n, insertion sort runs in $8n^2$ steps, while merge sort runs in $64n\log n$ steps. For which values of n does insertion sort beat merge sort?

For insertion sort to beat merge sort for a given size of n inputs then;

insertion_steps must be < merge_sort steps

Thus the inequality

$$8n^2 < 64n\log n$$

$$\frac{8n^2}{8n} < \frac{64n\log n}{8n}$$

$$n < 8\log n$$

Using Trial and Error Method.

Assumption; if $n = 10$ then $10 < 8$, ∴ the input size we are looking for has to be approximately less than 8 ifor the machine to beat its state of ~~using~~ merge sort using insertion sort

Taking $n = 5$, in $n < 8\log n$

$$5 < 8\log 5$$

$$5 < 5.5917$$

insertion sort beats merge sort for $n = 5$

Taking $n = 6$,

$$6 < 8\log 6$$

$$6 < 6.2252$$

insertion sort beats merge sort for $n = 6$

Taking $n = 7$,

$$7 < 8\log 7$$

$$7 \geq 6.7601$$

merge sort beats ~~merge~~ insertion sort for $n = 7$

Thus insertion sort beats merge sort for $2 \leq n \leq 6$. For $n \geq 7$ merge sort becomes faster.

2 Consider the following grading policy, specifically the rounding policy

A. Every student receives a grade or mark in the range of 0 to 100. No negatives are allowed.

B. Any grade below 50 is failure

C. If the difference between the grade or mark and the next multiple of 5 is less than 3, round grade or mark up to the next multiple of 5. eg if marks = 69 round to 70 (70 - 69 is less than 3)

D. If the value of is less than 50, no rounding occurs as the result will still be a failing grade e.g if mark = 43 is less than 50, no rounding

Write down your pseudo code to implemnt an algorithm that takes in a mark and implements the rules suggested above.

1 Define a function called rounding_grade that takes a list of marks.

2. Create an empty list called rounded_marks to store the final results.

3. For each mark in the list (marks), do the following:

a. If the mark is less than 0 or more than 100. Print/say "Invalid mark" along with the mark and skip to the next mark.

b. If the mark is below 50:

- Add the mark to the rounded_marks list without changing it.

4 For marks 50 ~~and not~~ or above:

a. Find the next multiple of 5 (eg for 69 is 70).

b. Check the difference between the mark and this next multiple of 5:

- If the difference is less than 3, round up to this ~~to~~ next multiple of 5

- else keep the original mark unchanged and add it to the rounded_marks list.

5 Return the rounded_marks list.