Rounding Rules
Point value 100/100

Suppose the local school district is adopting a new method of rounding grades. The following are the rules:

All grades will be in the range of 0 to 100.
The grading scale is as follows:

Grade	Range
A	100 – 85
B	84 – 70
C	69 – 55
D	54 – 40
F	39 - 0

The new rounding rules are as follows:

Rule 1:
If the difference between the grade and the **next** multiple of 5 is less than 3, round the grade up to the next multiple of 5. Ex. 81 is 4 points from 85 (the next multiple of 5) therefore the grade is not rounded. However, 88 is 2 points from 90, therefore, the grade will be rounded to 90.

Rule 2:
If the grade is less than 38 do not round (do nothing).

You will be given a set of grades that will be read from a file. Round the grades that should be rounded.
An example of an input file and the expected output is as follows:

Example input:

5
94
86
37
44
90

5 represents the number of grades to be read

The output for this input file would be the following:

2 = 0.40% of the students benefited from the grading policy change.
The new grades are:
95 86 37 45 90
The following is a graph that represents the grade distribution for each grade category.
A: * * *
B:
C:
D: *
F: *

The grades should be stored in an array. Since you do not know how many grades there will be until run-time you **MUST** dynamically allocate the memory for the array that will hold the grades. Do not forget to free the memory when done with it.

I will provide the driver that I used for my program. You will complete the functions listed in the functions.h file. These functions are based on how I completed this program. You may change the prototypes of these functions, however, you may not combine these functions. Functions should be short and accomplish small task. If you decide to change these functions and I feel that you have to many task in one function, I will subtract points. You must add header guards to the functions.h file. You will need to provide the appropriate comments above each function. The comments should follow the format described below:

Return: (if there is a return you are to explain what is being returned. If no return then put void.)

Parameters: (list the type of parameters and a short description of what it will be used for in the function.)

Description: (You are to write a detailed description of the functions purpose and how it will accomplish its purpose. In other words, describe what it is doing. It does not matter if the function is small and self-explanatory, you must provide a description.)

A substantial number of points will be deducted if you do not provide appropriate documentation.

The functions provided are described below:

int* readData(FILE*, int*);
Read the number of grades, allocated the memory, and read the data storing it in the allocated memory.

void determineGrade(int*, int);
For each grade in the array, determine if the grade should be rounded or not. If it should be rounded then change it in the array.
This is the function that I printed the statement saying how many students benefited from the new grading policy.

void printGrade(int, int*);
Prints the grades.

void CheckArguments(int argc);
Ensures the correct number of command line arguments are given. If not print:
To few command line arguments. Exiting program.
Then exit.

void checkFile(FILE*);
Check that the file pointer is not NULL. If it is NULL, print the following message:
The file did not open successfully. Exiting program.
Then exit.

void calculatePercent(int, int *);
Here you will need to determine how many A's, B's, C's,D's and F's there are in the new grades. It is here that I called printGraph for each grade category.

void printGraph(int p);
This function prints the *'s in the graph.

Make sure you follow all instructions. Points will be deducted if you do not follow directions.

Because leaving a file pointer unnecessarily open is a potential security risk, you **must** close the file pointer after you are finished with it.