Problem 0. Read the following Zybooks sections and solve the problems therein.
Sections 9.4 to 9.7, and 10.1 to 10.3 by 8pm Tuesday November 29th.
Sections 10.6, 10.9 to 10.16 by 8pm Thursday December 1st.

Problem 1.
Write a program recursive.c that takes as input an integer n and evaluates the function

\[
 f(n) = \begin{cases} 
 0 & n = 0, \\
 f(|n| - 1) + 6 & \text{if } n \neq 0 \text{ is even}, \\
 f(|n| - 1) - 5 & \text{if } n \text{ is odd}. 
\end{cases}
\]

The program prompts the user for a value until they wish to stop. It should use the recursive function:

\texttt{int evaluate(int)}

takes an integer \texttt{x} and returns \(f(x)\).
Problem 2. Write a program `median.c` that accepts several sequences and a single threshold—all of type `double`, calculates the median of each sequence, and displays the medians that are strictly smaller than the threshold, printing them to a 2-digit precision in non-decreasing order, and repeating medians shared between sequences. If no median is below the threshold the program prints an appropriate statement.

You can assume that there will be between one and fifty sequences, each of length between 1 and 100, and recall that the median of an even-length sequence is the average of the two middle elements.

```
(-) $ a.out
Number of sequences: 5
Sequence 1: 1.34 9.46 2.88
Sequence 2: 7.9 31
Sequence 3: 29 72 53.96
Sequence 4: 13 8.42 49 10 55 38 50
Sequence 5: 100 -299 45 -3.0
Threshold: 31
0.00 9.00 9.00 smaller than 31.00

(-) $ a.out
Number of sequences: 4
Sequence 1: 71 85 102.43
Sequence 2: -16 -89 -20
Sequence 3: 0 0 0
Sequence 4: 4.5 6.4 100 324
Threshold: 250
-20.00 0.00 53.20 78.00 smaller than 250.00

(-) $ a.out
Number of sequences: 1
Sequence 1: -234.987 12.456
Threshold: 500
234.00 smaller than 500.00

(-) $ a.out
Number of sequences: 2
Sequence 1: 1.34 9.46 2.88
Sequence 2: 7.9 31
Threshold: 0
No medians smaller than 0.00

(-) $
```
Problem 3. Write a program `speedtest.c` that measures a user’s speed of typing word sequences. The program consists of three files, all can be downloaded from the class’s homework page.

- **speedtest.h** that you should *not* modify. It contains: (a) two macros: `WORD_NUM` denotes the number of words in the “dictionary” where typed words are selected from, and `WORD_LEN` denotes the maximum length of a word in the dictionary; (b) forward declaration for a global string array `wordList` consisting of `WORD_LEN` words each of length at most `WORD_NUM` that describes the dictionary; (c) forward declarations of two functions you are asked to implement in the file `speedtestfunctions.c` described next.

- A file `speedtestfunctions.c`, consisting of the following two functions that you need to define:

  - `void get Sentence(int SLen, int randArray[])`: Called by `main` and takes as inputs the number `SLen` of words the user needs to type, and an integer array `randArray` that will store the indices (in `wordList`) of each of these words. The function calls `rand()` for `SLen` times, where call `i` generates a random number between 0 and `WORD_NUM-1` and stores it in `randArray[i]`. For example, if the second call was 4, then `randArray[1]` will be 4, indicating that the second word in the sentence will be `wordList[4]`, namely “picture” according to the current definition of `wordList` in `speedtest.c`.

  - `int match(int SLen, int randArray[], char arr2[])`: also called by `main`. Takes as input the number `SLen` of words to be typed, the array `randArray` containing the indices of these words, and array `arr2` that has the string entered by the user. It returns the number of word typed by the user that exactly match the requested words, for example, if `SLen` is 2, `randArray` is `{0, 3}`, and `arr2` is “dog first”, then `match` will return 1.

- A file `speedtest.c` that you need to modify and write `main()` and no other function. Main does the following:

  - Calls `srand` using the current time to initialize the random seed.
  - Calls `rand` to generate a random number between 1 and `WORD_NUM` that is the number of words the user should type, and initializes `SLen` accordingly.
  - Defines an array `randArray` of size `SLen` that will store the indices of the `SLen` words the user will need to type.
  - Prints the prompts, calls the function `getSentence`, and gets the user input.
  - Computes the time in seconds the user took to type the words.
  - Calls the function `match` and prints the number of correctly typed words and typing speed, defined as the number of correctly typed words per minute.
(-)$ a.out
Type: system cloudy unit cat cloudy
syste cloudy unit cat cloudy
You correctly typed 4 word(s) at 20.0 words per minute!
(-)$ a.out
Type: dog
dog
You correctly typed 1 word(s) at 30.0 words per minute!
(-)$ a.out
Type: question cloudy cat second system unit dog
question cloudy cat second system unit dog
You correctly typed 2 word(s) at 15.0 words per minute!
(-)$ a.out
Type: first dog cat second system talk question
first dog cat second system question
You correctly typed 5 word(s) at 21.4 words per minute!
(-)$ a.out
Type: cat cloudy question second picture picture cat dog
cat cloudy question
You correctly typed 3 word(s) at 45.0 words per minute!
(-)$