1. (7.1) Write a **function** that loops through and returns a list with every even number between two
   integers (inclusive). The arguments to the function will be *smaller_num* and *larger_num*.

   Examples:

   - output_even(37, 1050) → [38, 40, 42, ...1048, 1050],
   - output_even(1, 2000) → [2, 4, 6, ...1998, 2000],
   - output_even(50, 199) → [50, 52, 54, ...196, 198]

2. (7.2) Write a **function** that finds the largest even number in a list *numbers*. Return -1 if not found.
   You may **not** use the built-in functions *max*(), *min*(), *sort*(), or *sorted*().

   Examples:

   - largest_even([3, 7, 2, 1, 7, 9, 10, 13]) → 10,
   - largest_even([1, 3, 5, 7]) → -1,
   - largest_even([0, 19, 18973623]) → 0

3. (7.3) To train for an upcoming marathon, Johnny goes on one long-distance run each Saturday. He
   wants to track how often the number of miles he runs exceeds the previous Saturday. This is called a
   progress day. Write a **function** that takes in a list of miles run every Saturday and returns Johnny's
   total number of progress days.

   Examples:

   - progress_days([3, 4, 1, 2]) → 2, (Two progress days, day 2 since (4 > 3) and day 4 since (2 > 1))
   - progress_days([10, 11, 12, 9, 10]) → 3,
   - progress_days([6, 5, 4, 3, 2, 9]) → 1,
   - progress_days([9, 9]) → 0

1. (7.1) Write a **function** that loops through a word and returns a list with every other letter of the word starting with the **first** letter. The function will take a single argument *word* (a string representing the word to process).

   Examples:

   - skip_letter("counterattack") → ["c","u","t","r","t","a","c"]
   - skip_letter("banana sunday") → ["b","n","n","s","n","a"]

2. (7.2) Write a **function** that returns a list with the factors of a given integer. The argument of the function will be *num* (integer to find factors for).

   Examples:

   - find_factors(12) → [1, 2, 3, 4, 6, 12],
   - find_factors(17) → [1, 17],
   - find_factors(36) → [1, 2, 3, 4, 6, 9, 12, 18, 36]

3. (7.3) There's a great war between the even and odd numbers. Many numbers already lost their lives in this war and it's your task to end this. You have to determine which group sums larger: the evens or the odds. The larger group wins.

   Write a **function** that takes a list of integers named *numbers*, sums the even numbers and odd numbers separately, then returns which of the two sums is larger.

   Examples:

   - war_of_numbers([2, 8, 7, 5]) → "odds", (since 2 + 8 = 10, 7 + 5 = 12, odds is larger)
   - war_of_numbers([12, 90, 75, 1, 1]) → "evens", (12 + 90 = 102, 75 + 1 + 1 = 77, evens is larger)
   - war_of_numbers([2, 10, 22, 243]) → "odds"

1. (7.1) Write a **function** that loops through a word and returns a list with every other letter of the word starting with the **second** letter. The function will take a single argument *word* (a string representing the word to process).

   Examples:

   - skip_letter("counterattack") → ["o","n","e","a","t","c"]
   - skip_letter("banana sunday") → ["a","a","a","s","n","a"]

2. (7.2) Write a function named *add_lists* that takes two lists *lyst*1 and *lyst*2 and adds the first element in *lyst*1 with the first element in *lyst*2, the second element *lyst*1 with the second element *lyst*2, etc. Return a new list containing the corresponding sums of the list1 and list2. You may assume both lists have the same length.

   Examples:

   - add_lists([1, 3, 3, 1], [4, 3, 6, 1]) → [5, 6, 8, 2] (since 1+4=5; 3+3=6; 3+6=9; 1+1=2)
   - add_lists([1, 8, 5, 0, -7], [0, -7, 4, 2, -6]) → [1, 1, 9, 2, -13]
   - add_lists([1, 2], [-1, 1]) → [0, 3]

3. (7.3) To train for an upcoming marathon, Samuel goes on one long-distance run each Saturday. He wants to track how often the number of miles he runs fall short of the previous Saturday. This is called a lag day. Write a **function** that takes in a list of miles run every Saturday and returns Samuel's total number of lag days.

   Examples:

   - lag_days([5, 3, 2, 1]) → 3, (3 lag days, day2 since (3<5), day3 since (2<3), and day4 since (1<2))
   - lag_days([10, 11, 12, 9, 10]) → 1,
   - lag_days([6, 5, 4, 3, 2, 9]) → 4,
   - lag_days([9, 9]) → 0

1. (7.1) Write a **function** that loops through a word and returns a list with every other letter of the word starting with the **first** letter. The function will take a single argument *word* (a string representing the word to process).

   Examples:

   - skip_letter("counterattack") → ["c","u","t","r","t","a","c"]
   - skip_letter("banana sunday") → ["b","n","n","s","n","a"]

2. (7.2) Write a **function** that takes 3 numbers as arguments, num_1 (first number), num_2 (second number), and num_3 (third number). Return a list of the integers in ascending order. You may **not** use the built-in functions $max()$, $min()$, $sort()$, or $sorted()$.

   Examples:

   - ascending_order(2, 3, 1) → [1, 2, 3],
   - ascending_order(10, 1, 25) → [1, 10, 25],
   - ascending_order(2, 45, 4) → [2, 4, 45]

3. (7.3) To train for an upcoming marathon, Samuel goes on one long-distance run each Saturday. He wants to track how often the number of miles he runs fall short of the previous Saturday. This is called a lag day. Write a **function** that takes in a list of miles run every Saturday and returns Samuel's total number of lag days.

   Examples:

   - lag_days([5, 3, 2, 1]) → 3, (3 lag days, day2 since (3<5), day3 since (2<3), and day4 since (1<2))
   - lag_days([10, 11, 12, 9, 10]) → 1,
   - lag_days([6, 5, 4, 3, 2, 9]) → 4,
   - lag_days([9, 9]) → 0