HOMEWORK 5

Solve the following problems using loops. Try to avoid using Python built-in functions unless absolutely necessary.

Problem 1. Write a function filter_positive_even(numbers) that takes a list of integers and returns a new list containing only the positive even numbers.

```
filter_even([1, -2, -3, 4, 5, 6])
```

should return [4, 6]. Note that -2 does not work because it is negative.

Problem 2. Write a function named is_empty(a_list) that returns True if the list is empty and False otherwise.

Problem 3. Write a function named multiply_factor(a_list, factor) that returns a list where each element in the original list is multiplied by the given factor. For example

```
multiply_factor([1, 2, -3], 2)
```

should return [2, 4, -6].

Problem 4. Write a function named numeric_values(a_list) that takes a list as input and returns a new list with only the numeric elements. Numeric values include both integers and floating-point numbers. For example:

```
numeric_values(["1", "apple", 1, 1.2, -4])
```

should return $[1, 1.2, -4]$.

Problem 5. Write a function named remove_element(a_list, element) that takes a list and an element as input and returns a new list with all occurrences of that element removed. For example

```
removed_element([0, "test", 1, "apple", 0, 1.1], 0)
```

should return [“test”, 1, ”apple”, 1.1].

Problem 6. Write a function uppercase_strings(a_list) that takes a list of strings and returns a new list with all the strings in uppercase. For example

```
uppercase_strings(["Anne", "Ben", "David"])
```

should return [“ANNE”, “BEN”, “DAVID”].

Problem 7. Write a function last_names(full_names) that takes a list of full names and returns a new list containing all last names. Here, we assume that each full name consists of a first name followed by a last name, separated by a space. For example,

```
last_names(["Anne Nguyen", "David Hilbert", "Michael Jordan", "
    Alan Turing"])
```

should return [“Nguyen”, “Hilbert”, “Jordan”, ”Turing”]. All of these individuals are famous, except for the first one (at least for now!).

Problem 8. Write a function join_strings(a_list) that takes a list of strings and joins them into a single string, separated by spaces.

```
join_strings(["I", "Love", "Lake", "Forest", "College"])
```

should return “I Love Lake Forest College”.

Problem 9. Write a function longest_string(a_list) that takes a list of strings and returns the longest string. For example

```
longest_string(["apple", "banana", "mango"])
```

should return “banana”.

For the following problems, please use list comprehension to solve them. You don’t need to write a function unless the question asks for it.

Problem 10. Given a list of circle radii, create a new list containing the corresponding areas of the circles, rounded to one decimal place. For example, if the list is

```
radii = [1,2 ,3]
```

then, the corresponding areas should be [3.1, 12.6, 28.3].

Problem 11. From a list of integers, create a new list that includes only the odd numbers using list comprehension. For example, if the given list is [1, 2, 3, 4, 5, 6], then the new list should be [1, 3, 5].

Problem 12. From a list of strings, create a new list that contains the first letter of each string. For example, if the given list is [“Apple”, “Banana”, “Mango”], the new list should be [“A”, “B”, “M”].

Problem 13. Given a list of words, create a new list containing only those words that are shorter than 4 characters. For example, if the list is [“Ant”, “Dog”, “Lion”, ”Fish”]), the new list should be [“Ant”, “Dog”].