Question 1 - 15 points

Write a function `reverse_array` that satisfies these specs:

- It takes one argument, called `numbers`, that is an array of integers.
- The function should return a new array of integers, that contains the same elements as `numbers`, but in reverse order.
- The function should not modify its argument `numbers` in any way.

For example:

- If numbers = {10, 20, 30, 15, 25}, then `reverse_array(numbers)` should return an array with elements `{25, 15, 30, 20, 10}`
- If numbers = {10, 3, 2}, then `reverse_array(numbers)` should return an array with elements `{2, 3, 10}`

Question 2 - 15 points

Write a function `most_rs` that satisfies these specs:

- It takes one argument, called `strings`, that is an array of strings.
- The function should return the element of `strings` that contains the most occurrences of the letter R (both ‘r’ and ‘R’ should be counted).
- If multiple strings tie for the most occurrences of the letter R, your function can return any of those strings that tie.

For example:

- If strings = `{"February", "ROAR", "winter"}`, then `most_rs(strings)` should return either "February" or "ROAR", since they both have two R’s, whereas winter has one R.
- If strings = `{"miRRor", "train", "treasure", "carry"}`, then `most_rs(strings)` should return "mirror", which has 3 R’s, whereas the other words have 1 or 2 R’s.

Question 3 – 14 points

Write a function `column_sums` that satisfies these specs:

- It takes one argument, called `filename`, that is the name of a file. The file is a spreadsheet of integers. We do not know in advance how many rows and columns the spreadsheet contains. However, **we know that all rows have the same columns**, and **all values are integers**.
- The function should return an array of integers called `result`, such that result has as many elements as the columns of the spreadsheet, and result[i] is the sum of values in the i-th column of the spreadsheet.

For example: suppose that a file called hello.txt contains this text:

```
8,9,4,6
9,9,9,7
1,1,7,7
9,9,9,3
6,9,6,6
0,4,0,1
2,8,8,7
5,1,9,0
```

Then, column_sums("hello.txt") should return an array with values {40, 50, 52, 37}.

Question 4 – 14 points

Write a function `find_position` that satisfies these specs:

- It takes one argument, called `numbers`, that is an array of integers.
- The function should return the position of the first element of `numbers` that is greater than 100.
- If no element of `numbers` is greater than 100, the function should return -1.

For example:

- If numbers = `{10, 30, 200, 20}`, then `find_position(numbers)` should return 2, since 200 (the first element greater than 100) is at position 2.
- If numbers = `{10, 30, 20, 10, 200, 20, 415}`, then `find_position(numbers)` should return 4, since 200 (the first element greater than 100) is at position 4.
- If numbers = `{10, 30, 20}`, then `find_position(numbers)` should return -1, since no element is greater than 100.

Question 5 - 14 points

Write a function `sum_odd_numbers` that satisfies these specs:

- It takes one argument, called `numbers`, that is an array of integers.
- The function should return the sum of all elements of `numbers` that are odd numbers (i.e., that leave a remainder of 1 when divided by 2).

For example:

- If numbers = `{11, 30, 5, 3}`, then `sum_odd_numbers(numbers)` should return 19, since 11+5+3 = 19.
- If numbers = `{1, 5, 5, 2, 2, 1}`, then `sum_odd_numbers(numbers)` `sum_odd_numbers()` should return 12, since 1+5+5+1 = 12.

Question 6 - 14 points

Write a function `select_numbers` that satisfies these specs:

- It takes two arguments, called `numbers, positions`, that are both arrays of integers.
- The function should return an array called `result`, whose length is the same as that of `positions`, and such that: `result[i]` is the element of `numbers` located at position `positions[i]`.

For example:

- If numbers = `{11, 30, 5, 3}` and positions = `{2, 0}`, then `select_numbers(numbers, positions)` should return an array with elements `{5, 11}`, since 5 is the element at position 2 of `numbers`, and 11 is the element at position 0 of `numbers`.
- If numbers = `{11, 30, 5, 3}` and positions = `{1, 0, 1, 2}`, then `select_numbers(numbers, positions)` should return a result equal to `{30, 11, 30, 5}`.

Question 7 - 14 points

Write a function `check_subset` that satisfies these specs:

- It takes two arguments, called `values, set`, that are both arrays of integers.
- The function should return `true` if every single element of `values` is also an element of `set`, and should return `false` otherwise.

For example:

- If values = `{11, 30}` and set = `{2, 0, 30, 11, 30, 5}`, then `check_subset(values, set)` should return `true`, since both 11 and 30 are elements of the second argument.
- If values = `{30}` and set = `{2, 0, 30, 11, 30, 5}`, then `check_subset(values, set)` should return `true`, since 30 is an element of the second argument.
- If values = `{30, 11, 10, 0}` and set = `{2, 0, 30, 11, 30, 5}`, then `check_subset(values, set)` should return `false`, because 10 is not an element of the second argument.