1) Define a function `add_line_numbers` with an argument representing a filename. The function creates a new file that contains the same text as the input file, but each line starts with a line number + whitespace. The name of new file is the same as the input filename but starting with `n_`. Use the exceptions to check whether the file exists.
(1p)

—

Input:

```
text.txt:
He was a Northern journalist,
and it was
in the interest of his paper
```

Output:

```
n_text.txt:
1 He was a Northern journalist,
2 and it was
3 in the interest of his paper
```

****

2) Define a function `my_filtered_map` with arguments representing a list, a function `fun` (with one parameter), and the **keyworded arguments** `min` and `max`. On each number element in the list, call the function `fun` and save the results into the new list (use `list comprehension`. If the keyworded arguments are passed, filter the new list such that it contains only the values in the interval ⟨ `min`, `max` ⟩. For the testing, use an anonymous function that returns double of the original value. Do not use `map` function.
(2p)

—

Input:

```
my_filtered_map([1,2,3,"x",5,8,13], your_lambda_function, min=5, max=20)
my_filtered_map([True,-2.2,-1,0,1,2], your_lambda_function, max=0)
```

Output:

```
[6, 10, 16]
[-4.4, -2, 0]
```

****

3) Define a function `bank_account` that takes **variable-length** parameter list. Each file contains bank operations in the format:
`1001 D 500` # 500 is inserted into account 1001
`1001 W 500` # 500 is withdrawn from the account 1001
Go through the all files passed to the function and realize the operations. Return a dictionary containing the account number (key), and the account balance (value).
Use the function `split` to split the lines.
(2p)

—

Input:

```
bank_account("bank_01.txt", "bank_02.txt")
```

Output:

```
{1001: 1350.0, 1002: 1800.0}
```