# الاسم / اسلام السيد رمزى الغرباوى

# سيكشن / 1

# CHAPTER 3

**#Problem 1**

```
def remove_vowels(text):
    vowels = "aAeEiIoOuU"
    output = ''.join(list(filter(lambda x : x not in vowels , text)))
    print(f"Output String: {output}")
text = input("Enter a text: ")
remove_vowels(text)
```

**#Problem 2**

```
myList = [1,2,3,4,5,6,7,8,9,10]
```

```
oddNumbers = list(filter(lambda x : x % 2 == 1 , myList))
squaredOddNumbers = list(map(lambda x : x**2 , oddNumbers))

print(f"Squared Odd Numbers: {squaredOddNumbers}")
```

**#Problem 3**

```
def fibonacci(n):
  if n <= 1:
    return n
  return fibonacci(n-1) + fibonacci(n-2)
print(fibonacci(10))
```

**#Problem 4**

```
def make_adder(n):
    def adder(x):
        return n + x
    return adder
add_five = make_adder(5)
```

```
print(add_five(10))
```

**#Problem 5**

```
def apply_twice(func , number):
    return func(number)
print(apply_twice(lambda x : x + 1 , 5))
```

**#Problem 6**

```
from functools import reduce
from collections import Counter



data = [
"The quick brown fox",
"jumps over the lazy dog",
"Python is fun"
]
```

```
stopwords = {"the", "is", "a", "an", "over"}


words = reduce(
lambda acc, sentence: acc + sentence.lower().split(),
data,
[]
)


filtered_words = filter(lambda w: w not in stopwords, words)


freq = dict(Counter(filtered_words))
print(freq)
```

**#Problem 7**

```
def my_reduce(func, iterable, initializer=None):
    it = iter(iterable)
    if initializer is None:
```

```
        value = next(it)
    else:
        value = initializer
    for element in it:
        value = func(value, element)
    return value

numbers = [1, 2, 3, 4, 5]
print(my_reduce(lambda x, y: x + y, numbers))
# Output: 15
```

**#Problem 8**

```
def log_call(func):
    def wrapper(*args, **kwargs):
        print(f"Calling {func.name} with {args}, {kwargs}")
        result = func(*args, **kwargs)
        print(f"{func.name} returned {result}")
```

```
        return result
        return wrapper
@log_call
def multiply(a, b):
    return a * b
```

## MCQ:

9. **Always returns the same output for the same input**

10. **Immutability**

11. **map()**

12. **functools**

13. **An iterator containing elements that satisfy the condition**

# True or False:

14. True

15. False

16. True

17. False

18. True