Information Secutiry - Week 3

Arvid Lindstrom - s2740761, Nil Stolt Anso - s2705338,
Razvan Andrei Poinaru - s2914751

October 2, 2017

Abstract

Exercises submitted: 11, 12, 14, 15 and 17

Exercise 11

Program Output

```
formation Sec/infosecRepo/infosec2017/week3/ex11$ python repeatedSquares.py
Base: 43210 # <-- user input
Exponent: 23456
Modulus: 99987
Recursive solution = 82900
Iterative solution = 82900
```

Source of Program

```
from timeit import default_timer as timer

# Function used for reference since pythons
# pow() is using some optimizations
def naiveEponentiation(base, exponent):
    for i in range(exponent-1):
        base *= exponent
    return base

def repeatedSquaresIter(base, exponent, modulus):
    # Reference: Stamp's Book, page 99

    #1. Find exponent in binary
    binStr = bin(exponent)[2::] #skip the '0b'-part

    #2. Initialize values
    exponent = 0
    output = 0

    #3. Loop through binStr from MSB to LSB
    for bit in binStr:
        output = pow(pow(base, exponent), 2)
        #4. Calculate new exponent
        exponent = (exponent * 2)

        if(bit == '1'):
            output *= base
            exponent += 1
        output = (output % modulus)

    return output
```

```
def repeatedSquaresRec(base, exponent, modulus):
    # base case
    if(exponent == 1):
        return base % modulus
    # recursive step
    else:
        returned = repeatedSquaresRec(base, exponent / 2, modulus)
        returned = returned * returned % modulus
        if(exponent % 2 != 0):
            returned = returned * base % modulus
    return returned % modulus

base = int(raw_input("Base: "))
exp = int(raw_input("Exponent: "))
mod = int(raw_input("Modulus: "))

print "Recursive solution = " + \
    str(repeatedSquaresRec(base, exp, mod))
print "Iterative solution = " + \
    str(repeatedSquaresIter(base, exp, mod))
```

Exercise 12

Key used

The key: uoieazyxwvtsrqpnmlkjhgfdcb
Maps onto:
abcdefghijklmnopqrstuvwxyz

Decrypted text

```
9 common security awareness mistakes (and how to fix them)

To err is human, but to err in cyber security can cause major damage to an
organization. It will never be possible to be perfect, but major improvement
is possible, just by being aware of some of the most common mistakes and their
```

Source of Program

```
#!/usr/bin/python

import sys

def readFile(name):
    file = open(name,'r')
    return file.read()
```

Exercise 14

Key used

The key: uoieazyxwvtsrqpnmlkjhgfdcb
Maps onto:
abcdefghijklmnopqrstuvwxyz

Decrypted text

```
9 common security awareness mistakes (and how to fix them)
```

```
To err is human, but to err in cyber security can cause major damage to an
organization. It will never be possible to be perfect, but major improvement
is possible, just by being aware of some of the most common mistakes and their
```

Source of Program

```
#!/usr/bin/python

import sys

def readFile(name):
    file = open(name,'r')
    return file.read()
```

Exercise 15

Key used

The key: uoieazyxwvtsrqpnmlkjhgfdcb
Maps onto:
abcdefghijklmnopqrstuvwxyz

Decrypted text

```
9 common security awareness mistakes (and how to fix them)

To err is human, but to err in cyber security can cause major damage to an
organization. It will never be possible to be perfect, but major improvement
is possible, just by being aware of some of the most common mistakes and their
```

Source of Program

```
#!/usr/bin/python

import sys

def readFile(name):
    file = open(name,'r')
    return file.read()
```

Exercise 17

Key used

The key: uoieazyxwvtsrqpnmlkjhgfdcb
Maps onto:
abcdefghijklmnopqrstuvwxyz

Decrypted text

```
9 common security awareness mistakes (and how to fix them)

To err is human, but to err in cyber security can cause major damage to an
organization. It will never be possible to be perfect, but major improvement
is possible, just by being aware of some of the most common mistakes and their
```

Source of Program

```
#!/usr/bin/python

import sys

def readFile(name):
    file = open(name,'r')
    return file.read()
```