Assignment 1

1)

i) $f(n) = n^2 + 200$ $\qquad$ $g(n) = n^2 - 150$

$f = O(n^2)$ $\qquad$ $g = O(n^2)$

$f = g$, $\therefore$ $f = \Theta(g)$

ii) $f(n) = \log_{10} n$ $\qquad$ $g(n) = \log_2(n^3)$

$f = O(\log n)$ $\qquad$ $g = O(\log n)$

$f = g$, $\therefore$ $f = \Theta(g)$

iii) By property: For every $r > 1$ & $d > 0$, $n^d = O(r^n)$

$\therefore$ $g(n) = n^{1/3} = O(f(n) = 10^n)$

$g = O(f)$

$\therefore$ $f = \Omega(g)$

iv) $f(n) = 7n \log(n)$ $\qquad$ $g(n) = n^{1.25}$

$f = O(n \log n)$ $\qquad$ $g = O(n \cdot n^{1/4})$

Compare $\log n$ & $n^{1/4}$

By Big O property, $\log_b(n) = O(n^x)$ $\qquad$ $b > 1$, $x > 0$

$\therefore$ $f = O(g)$

v) $f(n) = n2^n$ $\quad g(n) = 3^n$

need C, N so that $n2^n \leqslant C3^n$ for $n \geqslant N$

$$n\left(\frac{2}{3}\right)^n \leqslant C$$

$C=5, N=5 \Rightarrow 5\left(\frac{2}{3}\right)^5 \leqslant 5$ ✓

$n\left(\frac{2}{3}\right)^n \to 0$ as $n \to \infty$

$\therefore f = O(g)$

vi) $f(n) = n$ $\quad g(n) = n^{1+\cos n}$ $\quad 0 \leqslant 1+\cos n \leqslant 2$

$f = O(n)$ $\quad g = O(n^2), \; g = \Omega(1)$

Due to oscillation, there is no tight bounds.
$\therefore$ none of the answers match.

vii) $f(n) = \log_2(n)$ $\quad g(n) = \log_{16}(n)$

$f = O(\log n)$ $\quad g = O(\log n)$

$f = g, \; \therefore f = \Theta(g)$

viii) $f(n) = 2^n$ $\quad g(n) = 2^{n+1} = 2 \cdot 2^n$

$f = O(2^n)$ $\quad g = O(2^n)$

$f = g, \; \therefore f = \Theta(g)$

2) $F_n = 0, 1, 1, 2, 3, 5, 8, 13, 21, 34$ $\quad$ $F_{n+1} = F_n + F_{n-1}$

$\qquad\qquad\qquad\qquad\qquad\quad F_6$

i) Base Case: $F_6 = 8$

$2^{0.5*6} = 2^3 = 8$

$\therefore F_n \geq 2^{0.5n}$ for $n=6$ ✓

Induction Hypothesis: $F_n \geq 2^{0.5n}$ $\qquad F_{n-1} \geq 2^{0.5(n-1)}$

$F_{n+1} \geq 2^{0.5n} + 2^{0.5(n-1)}$

$F_{n+1} \geq 2^{0.5n}(1 + 2^{-0.5})$

$F_{n+1} \geq 2^{0.5n}(\sim 1.7)$

$\hookrightarrow > 2^{0.5} = 1.414$

$F_{n+1} \geq 2^{0.5n}\, 2^{0.5}\, c \quad , \quad c > 1$

$F_{n+1} \geq c\, 2^{0.5(n+1)}$

$\therefore F_{n+1} \geq 2^{0.5(n+1)}$, induction hypothesis ✓

Therefore by induction, $F_n \geq 2^{0.5n}$, for all $n > 6$

ii) $F_n \leq 2^{cn}$, $n \geq 0$

$F_n \leq 2^{cn}$ for $c \in \mathbb{R}^+$, $n = 0, 1, 2$ (Since last two terms need to follow equation)

$F_{n+1} \leq 2^{cn}(1 + 2^{-c})$ $\quad$ From part i

$F_{n+1} \leq 2^{c(n+1)}\left(\frac{1+2^{-c}}{2^c}\right)$

$\frac{1+2^{-c}}{2^c} > 1$

$1 + 2^{-c} > 2^c$

$\frac{2^c+1}{2^c} > 2^c$

$0 > (2^c)^2 - 2^c - 1$

$\frac{1 \pm \sqrt{1+4}}{2} = \phi$ golden ratio

$2^c > \phi$

$c \geq \log_2 \phi$

$c \geq 0.6942$

$\therefore$ let $c = 0.7$

iii) Using part ii,

$c < \log_2 \phi$

$c < 0.6942\ldots\ldots$

$\therefore c = 0.694$ (rounded)

3)

i) $x = qN + r$

$x \bmod N = r = x' \bmod N$ $\Big\}$ $x = qN + x'$

$y = pN + r'$

$y \bmod N = r' = y' \bmod N$ $\Big\}$ $y = pN + y'$

$$xy = qpN^2 + x'pN + qy'N + x'y'$$

$$\xrightarrow{\bmod N} \cancel{qpN^2} + \cancel{x'pN} + \cancel{y'qN} + x'y'$$

$$= \boxed{x'y' \bmod N}$$

ii) $3^k \bmod 2 = (3 \cdot 3 \cdot 3 \cdot \ldots \cdot 3)(\bmod 2) = [(3 \bmod 2) \cdot (3 \bmod 2) \cdot (3 \bmod 2) \cdots] = (3 \bmod 2)^k = 1^k = 1$

$\hookrightarrow (x \cdot y \cdot z \cdot a \cdot b \cdot \ldots) \rightarrow (x' \cdot y' \cdot z' \cdot a' \cdot b' \cdot \ldots)(\bmod N)$

iii) $4^{500} (\bmod 17) = 4^{4 \cdot 125} (\bmod 17) = (4^4)^{125} (\bmod 17) = 256^{125} (\bmod 17) = (256 \bmod 17)^{125} = (1)^{125} = 1$

4)

```python
###############################################################################
# student info
#
# WatIAM username: r6sarkar
# Student number: 20894095
###############################################################################


def fib1(n):
    """An inefficient implementation of computing the n-th Fibonacci number"""
    if n == 0:
        return 0
    if n == 1:
        return 1
    return fib1(n-2) + fib1(n-1)

def fib2(n):
    """An efficient implementation of computing the n-th Fibonacci number"""
    if n == 0:
        return 0
    fib = [0] * (n+1)
    fib[0] = 0
    fib[1] = 1
    for i in range(2,n+1):
        fib[i] = fib[i-2] + fib[i-1]
    return fib[n]
```

5)

```python
################################################################################
# student info
#
# WatIAM username: r6sarkar
# Student number: 20894095
################################################################################


# part (i) for modular exponentiation -- fill in the code below
def modexp(x, y, N):
    if y == 0:
        return 1
    z = modexp(x, math.floor(y/2), N)
    if y % 2 == 0:  # y is even
        return (z * z) % N
    else:  # y is odd
        return (x * z * z) % N


# part (ii) for extended Euclid  -- fill in the code below
def extended_euclid(a, b):
    if b == 0:
        return (1, 0, a)
    (x_a, y_a, d) = extended_euclid(b, a % b)
    return (y_a, x_a-(math.floor(a/b)*y_a), d)
```