## Problem 3

a)

i)

Big Theta: f(n) = theta(g(n)) means that there exist a c\_1 and c\_2 such that there exist a N for all x bigger than N we have c\_1\*g(x) <= f(n) <= c\_2 \* g(x)

Big O: f(n) = O(g(n)) means that there exists a *c* such that there exists a *N* for all x bigger than N we have f(n) <= c \* g(x)

Big Omega: f(n) = Omega(g(n)) means that there exists a *c* such that there exists a *N* for all x bigger than N we have c \* g(x) <= f(n)

ii)

Yes. A = big theta(2^(3n)). Since for sufficiently large n we have 2^(3n) + n^2 > (1/8) \* 2^(3n) and also that 2^(3n) + n^2 > 8 \* 2^(3n).

b)

don’t think we have covered this.

## Problem 4

i)

FFT(a): // the input of the function should be a list of value

if (length(a) <= 1): return a

a\_odd = a[odd] \\ all the element in odd position

a\_even = a[even] \\ all the element in even position

b\_odd = FFT(a\_odd) \\ compute the FFT for odd a

b\_even = FFT(a\_even) \\ compute the FFT for even a

b = new list \\ a new list with all the value 0

for k in 0..N/2-1:

b[k] = b\_even[k] + k-th root of unity \* b\_odd[k]

b[k+N/2] = b\_even[k] - k-th root of unity \* b\_odd[k]

return b

Personally I think the question is too hard for an exam.

ii)

FFT([2,3,4,5]) -> FFT([3,5]) -> FFT([5]) -> [5]

| |

FFT([2,4]) |

->FFT([2])- [2] FFT([3])

->FFT([4])- [4] |

[3]

FFT([2,4]) -> [6,-2]

FFT([3,5]) -> [8,-2]

FFT([2,3,4,5]) -> [14, -2+2i, -2, -2-2i]