Homework 8: Q2

**Name:** Xinkai Lin, xinkaili

*Don’t forget to input your list of collaborators and sources on* ***AutoLab****.*

**Please submit this file as a PDF.**

# **Algorithm Details**

Input: a, n

If a == 0, return 0

If n == 0, return 1

If n == 1, return a

For (i=1; i<n; i++) do

a = a \* a

end for

return a

1. **Algorithm Idea**

To solve this problem, I will present a recursive divide and conquer algorithm. As the question descript, we have 2 inputs a and n, where n is the exponential of a, and both are non-negative integers. Which it will be the parameter for the functions. Inside this recursive function, first, it will check for the base case, if a is 0, then return 0, if n is 0, then return 1. Next, it will check if n is an even number or an odd number. If n is even, then it will call a recursion where we square the value of a, and reduce the size of n in half. If n is odd, we will do the same thing call the recursion but in addition we need to multiply another a to make it odd number.

1. **Algorithm Details**

Input: a, n

Power (a, n) do

If a == 0, return 0

If n == 0, return 1

If n%2 == 0, return power (a\*a, n/2)

Else return a \* power (a\*a, (n-1)/2)

end recursion

1. **Proof of Correctness Idea**

First of all, this algorithm will check for the base case where when a is equal to 0, it will return 0 since 0 will remind the same no matter the power, when n is equal to 0, it will return 1 since any number of its power is 0 will be 1. During each recursive call. When the exponent n is even, it will square the value of a with its power n in half, or . Since the property of the exponential, is same as \* , and \* is same as , which eventually is just going to be the same as In addition, when n is odd number, we just need to add multiply another a to it and reduce the n by 1, or a \* By doing this, we can resolve the function as the a \* \* where same as or and fix the awkward situation when we have an odd number. Therefore, this algorithm is correct.

1. **Runtime Analysis**

The runtime for this algorithm is O(log(n)). Since in the function, each if-statement only perform O(1) time. And each time when we have a recurrence, the size of n will reduce in half. Such as: n→ → → … → 1 → 0, n > 1. Therefore, the runtime for this algorithm is O(log(n)).