### P1. (10 pts) Proof by Induction (you must provide a proof using induction).

### Let F1 be the Fibonacci numbers as defined in Section 1.2 (assume F0=1, F1=1, and Fk+1 = Fk + Fk-1). Prove

### 

### If you want to type your answer, this notation can be written as 'Sigma i from 1 to N-2, F\_i = F\_N - 2'

* For N>= 1,



P2. (5 pts) (5 pts) An algorithm takes 1 ms for input size 100. How long will it take for input size 800 if the running time is the following (assume low-order terms are negligible):

* + Linear
  + O(N log2 N) ; the logarithm to the base 2
  + Quadratic
  + Cubic

P3. (10 pts) For each of the following program fragments, give an analysis of the running time (using Big-Theta)

i)

i = 1

while i < n\*n:

sum+=1

i += 3

ii)

i = 1

while i < n\*n:

sum+=1

i \*= 3

iii)

for i in range(n):

for j in range(i\*2, n\*\*3):

for k in range(j):

sum+=1

iv)

for i in range(n):

for j in range(i\*2, n\*\*3):

if j < i:

for k in range(j):

sum+=1

v)

k = 0

n = 5

if n > 10:

k = n

else:

for i in range(n):

for j in range(n):

k+=1

P4. (10 pts) What is the asymptotic complexity of the following functions? Justify your answer.

1. (3 pts)

def fun1(n):

    i = 0

    if (n > 1):

        fun1(n - 1)

    for i in range(n):

        print" \* ",end="")

**Recurrence Relation:**

**Complexity in Big-Oh: Show your process of finding the complexity from the recurrence relation.**

1. (3 pts)

def fun(a, b):

    if (b == 0):

        return 1

    if (b % 2 == 0):

        return fun(a\*a, b//2)

    return fun(a\*a, b//2)\*a

**Recurrence Relation:**

**Complexity in Big-Oh: Show your process of finding the complexity from the recurrence relation.**

### (4 pts)

def func(n,start,end,aux):

if(n==1):

print("Move disk 1 from",start,"to",end)

return

func(n-1,start,aux,end)

print("Move disk",n,"from",start,"to",end)

func(n-1,aux,end,start)

**Recurrence Relation:**

**Complexity in Big-Oh: Show your process of finding the complexity from the recurrence relation.**