Instructor Name: Siddhartha Nath
Email: s.nath@northeastern.edu

Northeastern University

Program Structure and Algorithms (INFO 6205)
Quiz #1 – SAMPLE SOLUTIONS 30 points

Student NAME:

Student ID:

Question 1 (8 points)**.** *Please find the $O(\cdot)$ complexity of growth for the following functions.*

(a) *(4 points)* $f(x) = 5x! + 4x^3 \log x$.

$f(x) = O(x!)$

(b) *(4 points)* $f(x) = 5x^6 - 4x^3 + 1$.

$f(x) = O(x^6)$

Question 2 (8 points)**.** *Please rank the following four functions based on their $O(\cdot)$ complexity of running time. The function that has the least complexity should be ranked 1. Please explain your answer to get full credit.*

$f_1(x) = 7\sqrt{x}$; $f_2(x) = x^3$; $f_3(x) = \log_2 x$; $f_4(x) = \sqrt[3]{x}$

$f_3(x) = Rank1$; $f_4(x) = Rank2$; $f_1(x) = Rank3$; $f_2(x) = Rank4$.

Question 3 (4 points). *You are given a numpy array A = np.array([[1. , 4. , 5.], [9. , 7. , 4.]]). Please state what the following commands will print*

(a) *(1 points) print(A − 2)*
[[−1. 2. 3.]
[7. 5. 2.]]

(b) *(1 points) print(A**2)*

[[1. 16. 25.]
[81. 49. 16.]]

(c) *(2 points) print(A[0, :-1])*
[1. 4.]

Question 4 (10 points). *In a postfix expression, ∗ an operator is written after its operands. That is, 2 + 3 is 2 3 + in postfix notation. The operations are performed in the order in which they are written (left to right).*

Suppose you are given a postfix expression such as 5 9 3 + 4 2 ∗ ∗ 7 + ∗. Please explain in English how you can use a stack or queue to evaluate a postfix expression as this. (the postfix expression is equivalent to 5 ∗ ((9 + 3) ∗ (4 ∗ 2) + 7) which evaluates to 515).

We can use a stack and follow these steps.

1. *Read the input one token at a time.*
2. *If it is an integer, push it on the stack.*
3. *If it is a binary operator, pop the top two elements from the stack, apply the operator, and push the result back on the stack.*

push(5); Stack: {5}
push(9); Stack: {5 9}}
push(3); Stack: {5 9 3}}
push(pop() + pop()); Stack: {5 12}
push(4); Stack: {5 12 4}
push(2); Stack: {5 12 4 2}
*push(pop() * pop()); Stack: {5 12 8}*
*push(pop() * pop()); Stack: {5 96}*
push(7); Stack: {5 96 7}
push(pop() + pop()); Stack: {5 103}
*push(pop() * pop()); Stack: {515}*
pop()