\_\_\_\_\_ /100

**CS 515 Exercise D02: Stack ADT Application**

**Names: \_\_\_\_\_\_\_\_\_Liam Warren\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Wildcats ID’s :** 917694086

**Lecture Section: 01 / 02 (circle one)**

**Part I:**

Trace the following algorithms on the inputs provided to fill the result array S.

**Algorithm *spans1*(*X, n*)**

0 1 2 3 4

**Input:** array ***X*** of ***n*** integers

6 3 4 5 2

***n***

5

***X***

**Output:** array ***S*** of spans of ***X***

0 1 2 3 4

***S*** ← new array of ***n*** integers

3

2

***s***

***i***

1

1

1

***S***

for ***i*** ← 0 to ***n*** − 1 do

***s*** ← 1

while ***s***≤ *i*  and ***X***[***i***- ***s***] ≤ ***X***[***i***] 0 1

***s*** ← ***s*** + 1 1 1 2 2

***S***[***i***] ← ***s*** 3 3

return ***S***  4 1

0 1 2 3 4 5 6 7

**Algorithm** ***spans2***(***X, n***)

***S***

***X***

**Input:** array ***X*** of ***n*** integers

6 3 4 1 2 3 5 4

8

***n***

***i***

**Output:** array ***S*** of spans of ***X***

0 1 2 3 4 5 6 7

***S*** ← new array of ***n*** integers

1

2

1

1

2

1

3

6

***A*** ← new empty stack

for ***i*** ← 0 to ***n*** − 1 do

while ( ! ***A***.*empty*() and ***X***[***A***.*top*()] ≤ ***X***[***i***] ) do

***A****.pop*()

if ***A***.*empty*() then

***S***[***i***] ← ***i***+1

else

***S***[***i***] ← ***i***- ***A****.top*()

***A***

***A***.*push*(***i***) 7,6,0

return ***S***

**Part II:**

Analyze the following algorithms to determine their overall time complexity (“Big-Oh”).

**Algorithm *spans1*(*X, n*)**

**Input:** array ***X*** of ***n*** integers

**Output:** array ***S*** of spans of ***X***

***S*** ← new array of ***n*** integers

for ***i*** ← 0 to ***n*** − 1 do

***s*** ← 1

while ***s***≤ *i*  and ***X***[***i***- ***s***] ≤ ***X***[***i***]

***s*** ← ***s*** + 1

***S***[***i***] ← ***s***

return ***S***

Answer: 0(n2)

**Algorithm** ***spans2***(***X, n***)

**Input:** array ***X*** of ***n*** integers

**Output:** array ***S*** of spans of ***X***

***S*** ← new array of ***n*** integers

***A*** ← new empty stack

for ***i*** ← 0 to ***n*** − 1 do

while ( ! ***A***.*empty*() and ***X***[***A***.*top*()] ≤ ***X***[***i***] ) do

***A****.pop*()

if ***A***.*empty*() then

***S***[***i***] ← ***i***+1

else

***S***[***i***] ← ***i***- ***A****.top*()

***A***.*push*(***i***)

return ***S***

Answer: O(n)