ECS32B Midterm Exam

Geoffrey Mohn

TOTAL POINTS

62 / 100

QUESTION 1

Question 1 10 pts

1.1 A: n = 2000 **5 / 5**

✓ **- 0 pts** *Correct answer: 8 seconds*

1.2 B: n = 200,000 **0 / 5**

✓ **- 5 pts** *Incorrect result*

QUESTION 2

2 Question 2 **3 / 15**

✓ **- 3 pts** *O(n^2) not circled*

✓ **- 3 pts** *Omega(log n) not circled*

✓ **- 3 pts** *Incorrectly circled O(log n)*

✓ **- 3 pts** *Incorrectly circled Omega(n^2)*

QUESTION 3

3 Question 3 **10 / 10**

✓ **- 0 pts** *Correct*

QUESTION 4

4 Question 4 **6 / 10**

✓ **- 4 pts** *reversed front with rear of queue*

QUESTION 5

5 Question 5 **8 / 10**

✓ **- 2 pts** *Incorrect or empty stack contents for third input*

QUESTION 6

Question 6 15 pts

6.1 A: Convert to RPN **7 / 7**

✓ **- 0 pts** *Correct*

6.2 B: Evaluate RPN **8 / 8**

✓ **- 0 pts** *Correct*

QUESTION 7

7 Question 7 **8 / 15**

✓ **- 1 pts** *minor syntax error in peek implementation*

✓ **- 6 pts** *pop not implemented or implementation is semantically incorrect*

peek: return self.head.getData()

QUESTION 8

Question 8 15 pts

8.1 A: Call Tree **7 / 7**

✓ **- 0 pts** *Correct*

8.2 B: O(n) implementation **0 / 8**

✓ **- 8 pts** *Incorrect. New procedure is not O(n) or better or it is not semantically correct*

Name Geoffrey Mohn

ECS32B Spring 2019 Midterm Exam

Your Name	Mohn, Geoffrey
Student ID	912568148
Seat Row and Number	H 106

- ❖ We will hand out the exams first, this will take a while. Don't open the exams until we announce "You may begin".
- ❖ You will need to enter your name, student ID, and seat number on the test.
- ❖ When we call time you must stop writing immediately.
- ❖ If you fail to do the above items, it's a 10% demerit on the exam.
- ❖ Once begun, no talking, or use of computing devices, such as cellphone, smartphone, laptop, iPad, or calculator.
- ❖ Only two pages of notes are allowed, more than two will be considered cheating.
- ❖ Once begun, No Roving Eyes.
- ❖ If you have a question, step out of your seat, approach a TA or Instructor, and ask your question in a quiet, low voice.
- ❖ Violations of the above rules are reported to SJA without fail for adjudication.
- ❖ If you understand the homeworks and the lecture slides, this shouldn't be that hard.

Name Geoffrey [illegible]

Question 1 (10 points)

Suppose you are using a particular $O(n^2)$ algorithm and and your 1 Ghz computer it takes 2 seconds to solve a problem of size 1000.

2 seconds Problem size 1000 n^2

a) Estimate how long in seconds it will take to solve a problem of size 2000?

~~$\frac{\cancel{K} \times 2000^2}{\cancel{K} \times 1000^2} = \frac{x \text{ seconds}}{2 \text{ seconds}}$~~

~~$2 \times 4 = \frac{x}{2} \times 2$~~

~~$8 = x$~~

~~$\boxed{8 \text{ Seconds}}$~~

$\frac{K \times 2000^2}{K \times 1000^2} = \frac{x \text{ Seconds}}{2 \text{ Seconds}}$

$2 \times \frac{4}{1} = \frac{x}{2} \times 2$

$\boxed{8 \text{ Seconds}}$

b) Estimate how long in seconds it will take to solve a problem of size 200,000 ?

$\frac{\cancel{K} \times 200{,}000^2}{\cancel{K} \times 1000^2} = \frac{x \text{ seconds}}{2 \text{ seconds}}$

$400 = \frac{x}{2}$

$\boxed{800 \text{ Seconds}}$

Question 2 (15 points)

Circle all that are true

T(n) = 3n is

a) [✓ $O(\log n)$] [✓ $O(n)$] $O(n^2)$

b) $\Omega(\log n)$ [✓ $\Omega(n)$] [✓ $\Omega(n^2)$]

c) $\Theta(\log n)$ [✓ $\Theta(n)$] $\Theta(n^2)$

Name Geoffrey [illegible]

Question 3 (10 points)

Suppose we perform the following sequence of operations (as defined in your textbook) on a stack which is initially empty. What are the stack contents at the end of the sequence of operations and what number will be printed?

```
s = Stack()
s.push(42)
s.push(19)
s.push(88)
x = s.pop()
s.push(11)
s.pop()
s.peek()
x = s.peek()
print(x)
```

42

42, 19

42, 19, 88

42, 19

42, 19, 11

42, 19

$\boxed{19}$

Question 4 (10 points)

Below are the contents of a Python list used to implement the Queue() ADT as described in our book with the front of the queue at the end of the list. Assign an order to the expressions, beginning with the empty list, and add method names with arguments so that the transcript makes sense.

Order	Statement	Built in Python list contents
1	D = Queue()	[]
4	d.enqueue('Q')	['V','A','Q']
6	d.~~de~~ queue()	['Q']
3	d.enqueue('A')	['V','A']
5	d.dequeue()	['A','Q']
2	d.enqueue('V')	['V']

Name [illegible]

Question 5 (10 points)

Using the proper parenthesis procedure discussed in class check the following expression until the procedure balanced terminates. Show the stack contents at the end of execution as a Python list with the top at the end.

```
from Stack import Stack

def balanced(inStr):
    s = Stack()
    for symbol in inStr:
        if symbol == "(":
            s.push(symbol)
        else:
            if s.isEmpty():
                return False
            else:
                s.pop()

    if s.isEmpty():
        return True
    else:
        return False
```

Input	Return value	Stack contents as a Python list just before the function `balanced()` returns.
'()((()'	False	['(', '(']
'()()()'	True	[ ]
'()))'	False	[')', ')']

Name Geoffrey [illegible]

Question 6 (15 points)

a) Convert the following expression from infix to postfix so that it can be evaluated using an RPN calculator.

2 * 3 + 3 * (1 + 4)   6 + 15

((2×3)+(3×(1+4)))

((23×)(3(14+)×)+)

23× 314+×+

b) Evaluate scanning the postfix expression from left to right. Next to each operator or operand scanned, show the contents of the stack represented as a Python list, with the top at the end of the list, after the operator or operand is processed.

Operator or Operand	Stack
2	[2]
3	[2,3]
×	[6]
3	[6,3]
1	[6,3,1]
4	[6,3,1,4]
+	[6,3,5]
×	[6,15]
+	[21]

Name 99 Geoffrey Moln

Question 7 (15 points)
The following class uses a linked list to implement a stack. The push, pop, and peek methods are not fully implemented. In the space below the existing code complete the class definition to include complete implementations of these three missing methods.

```
class Node:
    def __init__(self,data):
        self.data = data
        self.next = None
    def getData(self):
        return self.data
    def getNext(self):
        return self.next
    def setData(self,newdata):
        self.data = newdata
    def setNext(self,newnext):
        self.next = newnext
class Stack:
    def __init__(self):
            self.head = None
    def isEmpty(self):
            return self.head==None
    def push(self, data):
            temp = Node(data)
            temp.setNext(self.head)
            Self.head= temp
    def Pop(Self):
            Self.head=None
    def PeeK(self)
        return Self.head
```

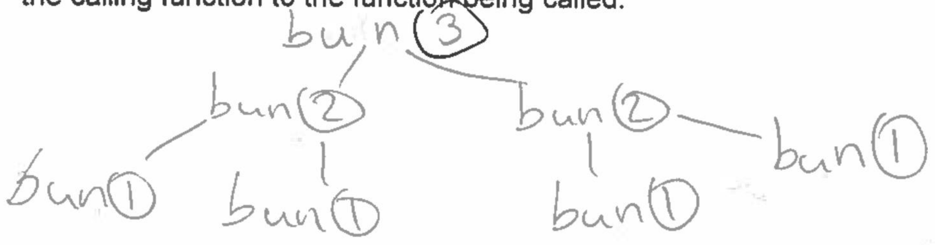
Name Geoffrey Mohn

Question 8 (15 points)

The recursive function bun(n) models the exponential growth of a population of bunnies. We assume in generation 1 we have one pair of adult bunnies. In generation n > 1, every pair of bunnies from the previous generation reproduces to make a new pair of bunnies. Also, bunnies never die!

```
def bun(n):
     if n == 1:
          return 2  # start with one pair
     else
          return bun(n-1) + bun(n-1) # previous pairs + new pairs
```

a) Draw a recursion tree diagram of the function calls to evaluate `bun(3)` where each call is a circle with the value of n in it, and if one function call results in another there is an arrow from the calling function to the function being called.



b) Write a faster version of bun below. You may use any trick to make it faster, as long as it's O(n) complexity or better.

```
def bun(n):
    bunny = 1
     for i in range(n):
          if n==1:
               return 2
          else:
          bunny = i x 2 x bunny
     return bunny
```