CMPE/CISC-365*
Test #1
September 27, 2019

Student Number (Required) ______________________

Name (Optional)________________________________

This is a closed book test. You may refer to one 8.5 x 11 data sheet.

This is a 50 minute test.

Please write your answers in ink. Pencil answers will be marked, **but will not be re-marked under any circumstances.**

The test will be marked out of 50.

Question 1	/12
Question 2	/12
Question 3	/12
Question 4	/12
Question 5	/2
TOTAL	/50

Divide et impera.

Attributed to Julius Caesar

QUESTION 1 (12 Marks)

State the "Big O" order class for each of the following algorithms. In all cases you may assume that the parameter n is an integer.

For each algorithm, give a justification of your answer.

(a)

```
def Q1.1(n):
    if n > 10:
        for i = 1 to n:
            for j = 1 to n:
                for k = 1 to j:
                    print i+j+k
    else:
        for i = 1 to 1000000:
            print i
```

Solution: O(n^3)

Explanation: For Big O classification we are only concerned about what happens when n is large so the else clause is not relevant. In the if clause the three nested loops give O(n^3)

Marking:

correct complexity -	2 marks
almost correct complexity (eg O(n^2)) -	1 mark
no answer or very wrong answer -	0 marks

explanation that shows understanding -	2 marks
explanation that shows limited understanding -	1 mark
none or very wrong explanation -	0 marks

(b)

```
def Q1.2(n):
    if n < 1:
        print n
    else if n <= 10:
        print n
        Q1.2(n-1)
    else:
        for i = 1 to 1000000:
            print i
```

Solution: O(1)

Explanation: for all $n > 10$, the for loop is activated and it always executes the same number of times.

Marking:

as for part (a)

(c)

```
def Q1.3(n):
    if n <= 1:
        print n
    else if n is odd:
        print n
        Q1.3(n+1)
    else:
        print n
        Q1.3(n/2)
```

Solution: O(log n)

Explanation:

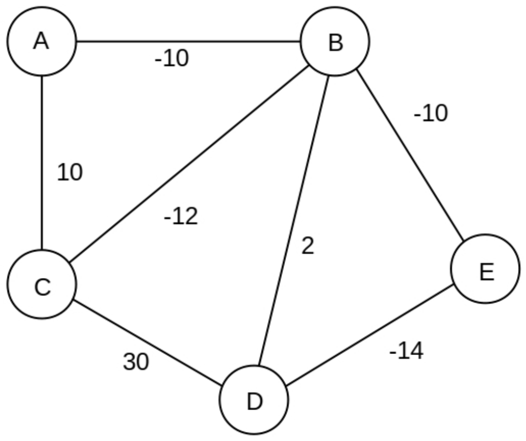
Basically the recursion goes from n to $\left\lceil \frac{n}{2} \right\rceil$ *in at most two steps. This is one of the standard recurrence patterns.*

Marking:

as for part (a)

QUESTION 2 (12 Marks)

Consider this graph, with edge weights as shown. Explain clearly why Dijkstra's Algorithm will fail to find the least-weight path from vertex A to vertex E. Where exactly does the algorithm go wrong?



Solution:
The first thing that happens is B is chosen, with a cost of -10 (path A-B)
Next C is chosen with a cost of -22 (path A-B-C)
Next E is chosen with a cost of -20 (path A-B-E)
This is incorrect: the least cost path from A to E is A-B-D-E with a cost of -22

Marking:

demonstrate clear understanding of how the algorithm works	**6 marks**
demonstrate partial understanding of how the algorithm works	**3 marks**
demonstrate significant confusion regarding the algorithm	**1 mark**

correctly identify the error made by the algorithm	**6**
almost get it right	**4**
way off	**1**

QUESTION 3 (12 marks)

Let A be a one-dimensional array of length n ($n \geq 1$), containing only 0′s and 1′s. All of the 0′s are to the left of all the 1′s. A may contain all 0′s or all 1′s.

Some examples of A are [0,0,0,0,0,0,1,1] , [1,1] , [0] , [0,1,1,1,1,1,1,1,1,1]

Create an algorithm to determine the number of 0′s in A. Your algorithm must run in O($n * \log n$) time.

Write your algorithm in clear pseudo-code or in a standard programming language.

After your algorithm, explain how you know it runs in O($n * \log n$) time.

*Solution: This question contains an error – the target complexity should have been O(log n). Stated as O(n*log n) the problem is trivially easy.*

*Any algorithm that runs in O(n*log n) or better is fine. The one I had in mind was to apply binary search. If you hit a 1, go left; if you hit a 0, go right. We know binary search runs in O(log n) time.*

Marking:

Some students will have taken an O(log n) or O(n) algorithm and added extra work to bring the complexity up to O(n*log n) as stated in the question. That's fine – they should get full marks

O(log n) or O(n) or (n*log n) Algorithm –	**6 marks**
Algorithm that doesn't work or O(n^2) or higher order algorithm –	**2 marks**

Reasonable Explanation	**– 6 marks**
Weak Explanation	**- 3 marks**
No Explanation or Very Incorrect explanation	**- 1 mark**

QUESTION 4 (12 Marks)

We have studied the Pair-Sum Algorithm. This question addresses finding **three** values that sum to a target figure.

Let A, B and C be 1-dimensional integers arrays of length n, and let k be any integer.

Create an algorithm to answer the question "Are there three values a, b and c such that a is in A, b is in B, c is in C, and a+b+c = k?"

For full marks your algorithm must run in O(n^2) time or better

You may assume that *sort()* is a built-in function that runs in O($n * \log n$) time

Solution:

```
Sort A
Sort B
for c in C:
    Pair-Sum(A,B,k-c)
    if a suitable pair of values a from A and b from B are found:
        print a, b, c
        stop
print "no solution"
```

*Discussion: The sorts run in O(n*log n). The loop iterates n times, and each iteration calls Pair_Sum which is an O(n) function. This gives us O(n^2)*

Marking:

There are other ways to achieve the same goal, but all the ones I know are variations on the idea used in the solution shown above. Some students will want to sort all three sets as a preliminary – that's fine.

Students may also try to accelerate the process by deleting all elements of the three sets that are > the target value. This is ok but in the worst case none of the values would be eliminated so it doesn't improve the complexity class.

Algorithm that finds the solution in O(n^2) time - 6 marks
Algorithm that finds the solution in O($n^2 * \log n$) time - 3 marks
(example of such an algorithm:

```
sort A
for each b in B
    for each c in C
        use binary search to see if k – (b+c)
                                  is in A
```

)

Algorithm that fails to find the solution
but runs in O(n^2) time - 2 marks
Algorithm that neither finds the correct solution
nor runs in O(n^2) time - 1 marks

Reasonable Explanation - 6 marks
Weak Explanation - 3 marks
No Explanation or Very Incorrect explanation - 1 mark

QUESTION 5 (2 Marks)

True or false:

MergeSort is named after Edouard Merge, a hermit who lived on a tiny tropical island and spent all his time sorting coconuts.

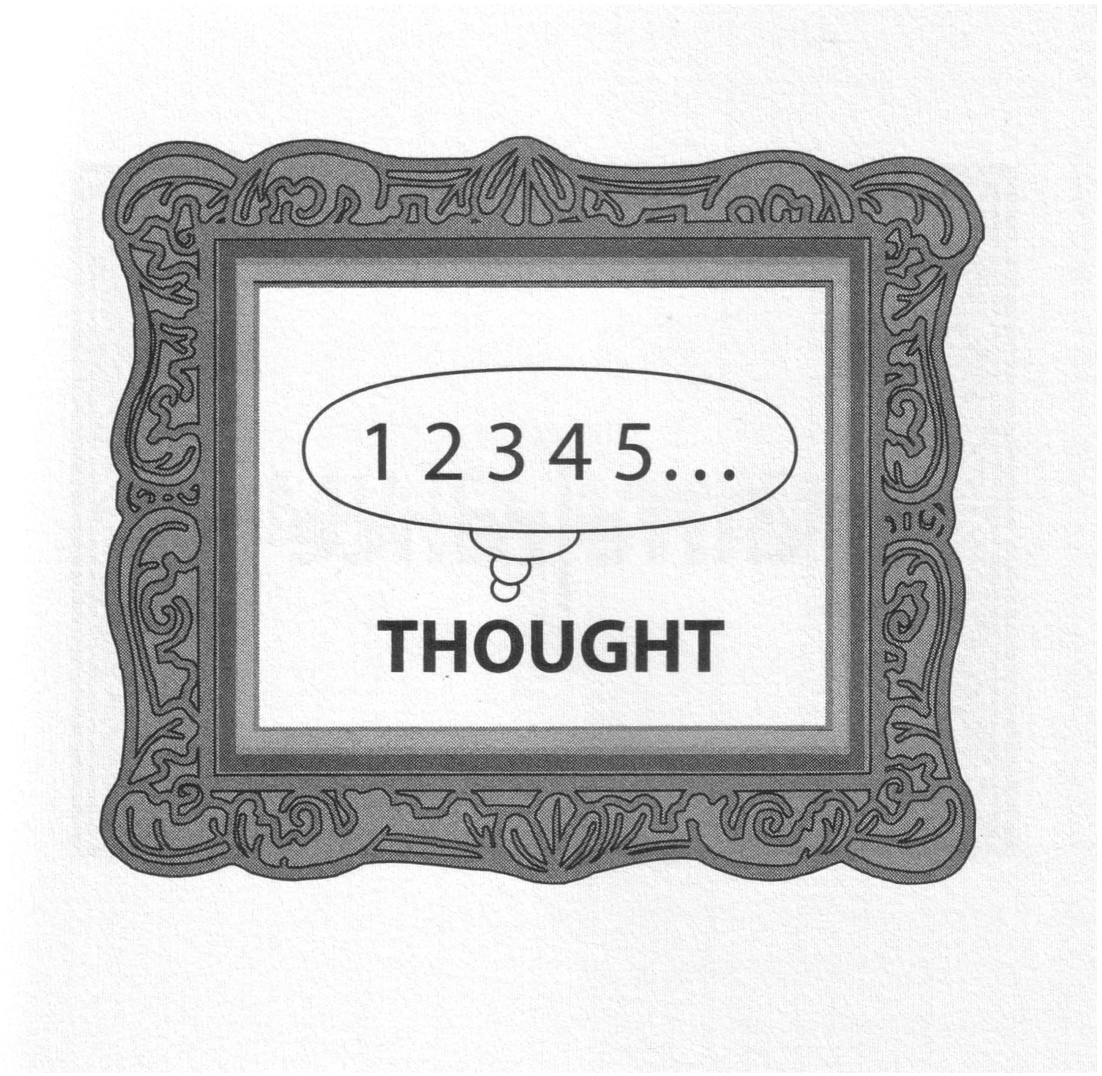
True FALSE

Solution: False

Marking:

Correct answer -	**2 points**
Incorrect answer -	**0 points**

Special Bonus Question: (0 marks)



What is the meaning of the figure above?