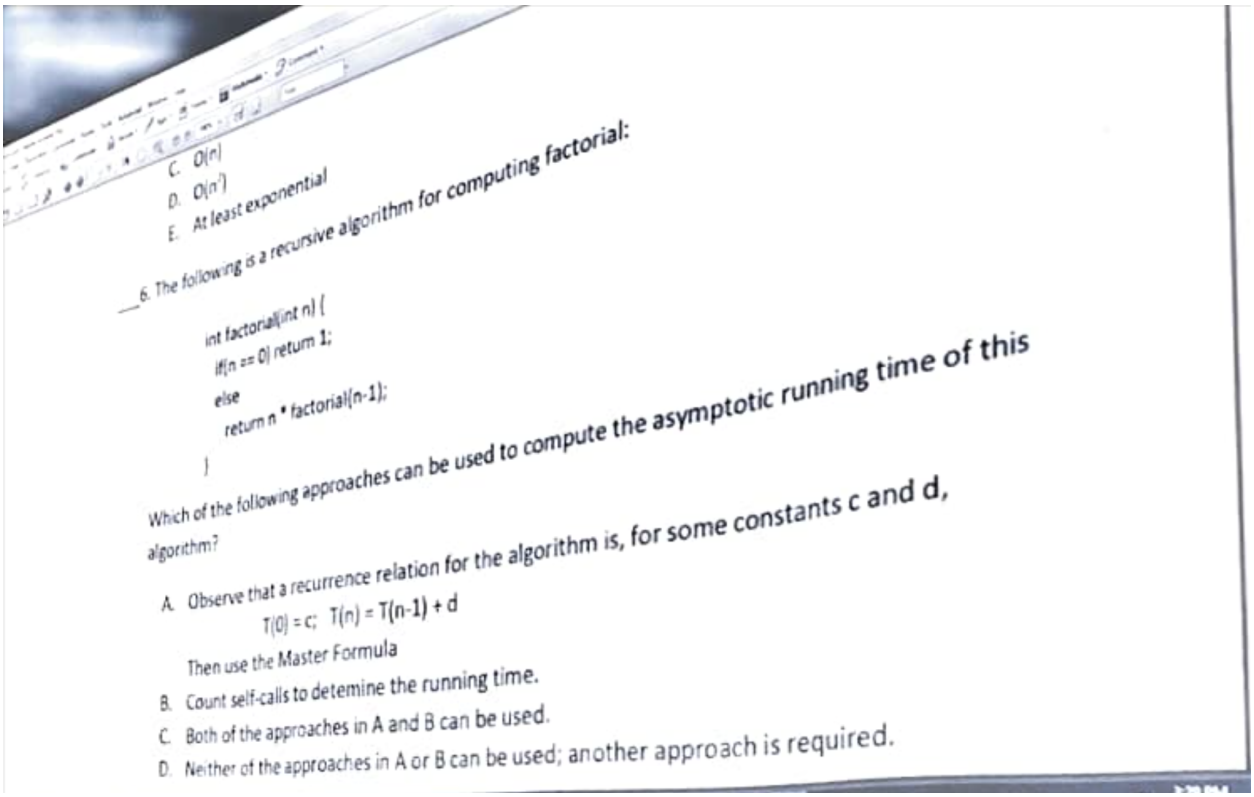
Multiple choice and True/False (25%)



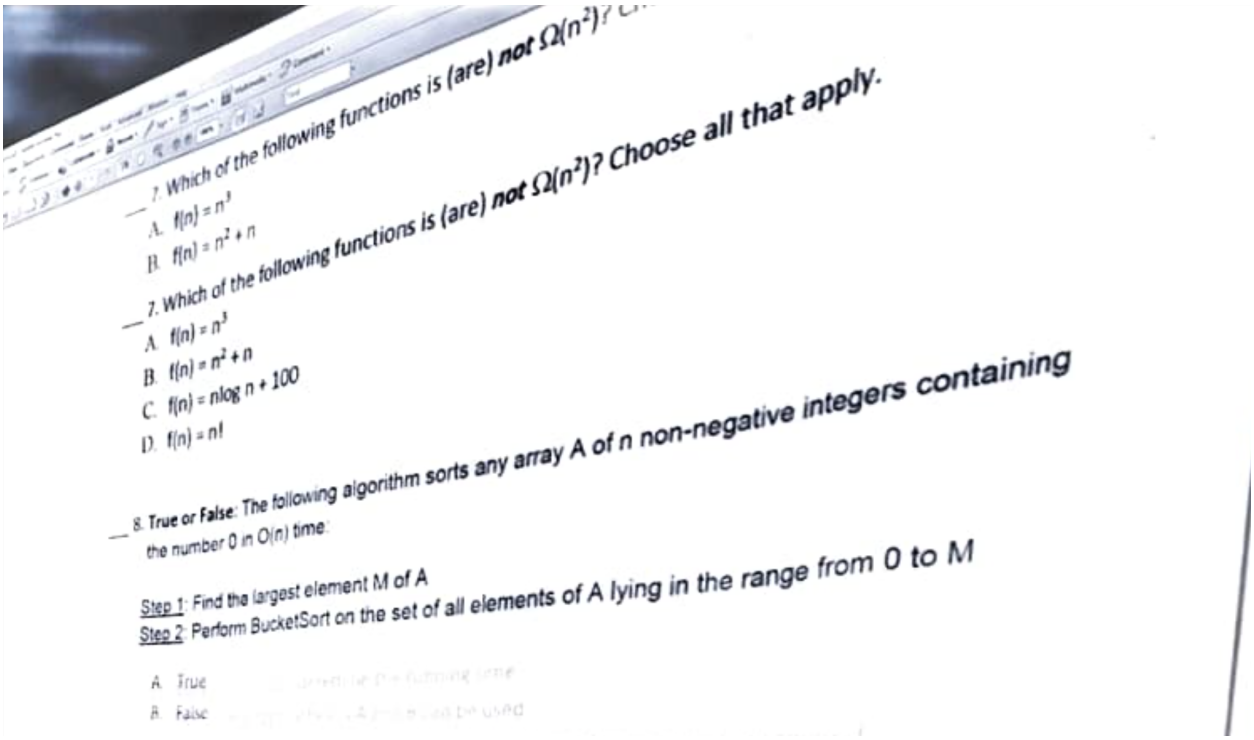
C. $O(n)$
D. $O(n^2)$
E. At least exponential

__6. The following is a recursive algorithm for computing factorial:

```
int factorial(int n) {
 if(n == 0) return 1;
 else
  return n * factorial(n-1);
}
```

Which of the following approaches can be used to compute the asymptotic running time of this algorithm?

A. Observe that a recurrence relation for the algorithm is, for some constants c and d,
$T(0) = c$; $T(n) = T(n-1) + d$
Then use the Master Formula
B. Count self-calls to detemine the running time.
C. Both of the approaches in A and B can be used.
D. Neither of the approaches in A or B can be used; another approach is required.



__7. Which of the following functions is (are) **not** $\Omega(n^2)$? Choose all that apply.

A. $f(n) = n^3$
B. $f(n) = n^2 + n$
C. $f(n) = n\log n + 100$
D. $f(n) = n!$

__8. **True or False**: The following algorithm sorts any array A of n non-negative integers containing the number 0 in $O(n)$ time:

Step 1: Find the largest element M of A
Step 2: Perform BucketSort on the set of all elements of A lying in the range from 0 to M

A. True
B. False

__**6.** The following is a recursive algorithms for computing factorial

```
Int factorial(int n) {
If(n==0) return 1;
Else
 Return n * factorial(n-1);
}
```

Which of the following approaches can be used to compute the asymptotic running time of this algorithm?
A. Observer that a recurrence relation for the algorithms is, for some constants c and d,

T(0) = c; T(n) = T(n-1) + d

Then use the Master Formula
B. Count self-calls to determine the running time.
C. Both of the approaches in A and B can be used.
D. Neither of the approaches in A or B can be used; another approach is required.

True/False

___ **8.**Ture or False: The following algorithms sorts any array A of n non-negative integers containing the number 0 in O(n) time:

Step1: Find the largest elemetns M of A
Step2: Perform BucketSort on the set of all elements of A lying in the range from 0 to M

A. True
B. False

<u>Short Answer (25%)</u>

2.(3points) Use the Master Formula to solve the following recurrence relation:
T(1) = 2; T(n) = 3T(n/3) + 4n

3.(4points) Formulate a recurrence relation for the running time of the following algorithm not-solve your recurrence relation). Write your answer in the box provided.

Algorithm recurSum1(n)

Input: a non-negative integer n

Output: the sum of n + n/2 + n/4 + …. + 1

If(n=0 || n=1) then return n

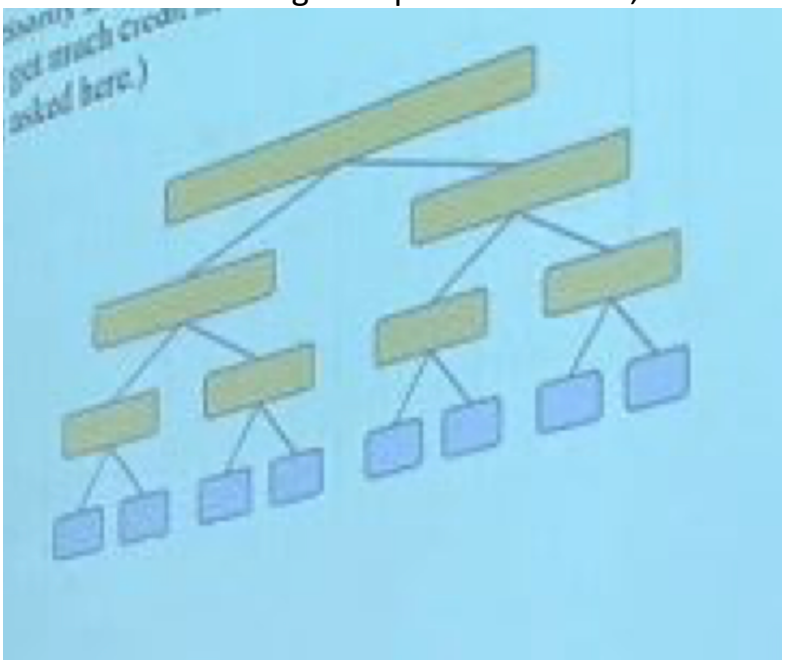
Return n+recurSum1(n/2)

⇨ T(0) = ? T(n) = ?

4. (3points) True or False: There is a comparasion-based sorting algorithm which, when run on

<u>Long Answer (50%)</u>

1. (6poinst) The diagram below shows a MergeSort recursion tree. Prove that the MergeSort recursion tree, representing the behaviour of MergeSort when running on input list of size n, necessary has height O(log n)



2. (6poinst) Use a QuickSort recursion tree to illustrate how QuickSort will sort[4,8,5,6,3,2,9]. Assume than, as QuickSort consider each sublist, it will use the

first element of the sublist as pivot. (If you use some other element for pivots, you will lose points).

3. (6points) The following algorithms determines whether a list of integers is in sorted order. Algorithms inSorted(List list) (Not complete question)

Input: a non-nulll list of integer

Output: true if the list is in sorted order, else false

If(…. <2) the return true

….

Output: The R-number R_n

If(n =0 || n=1 ||n=2) then

Return n+1

Return Rnum(n-3) * (Rnum(n-1) * Rnum(n-2)+1)

A.(3points) Prove that Rum is correct(that is, show that Rnum is a valid recursion and, for each nonnegative integer n, Rum(n) output R_n).

4. (6points) The R-Numbers are defined as follow: (Not complete question)

R_0 =1, R_1 =2, R_2 =3, R_0= R_{n-1} * (R_{n-1} + R_{n-2} + 1) (am not so sure with this, image not clear)

B.(3points) Prove that the running time for Rnum(n) is at least exponential. (You do not need to prove facts that were already proven in class)

5.(6points) An algorithms solves a problem by diving it into 3 sub-problems. Each recursive call divides the problems into one-third of the size of the problem; diving the problem in this way takes linear time each time. After solving all sub-problem, it combines the solutions in linear time.

A. (3 points) Give a recurrence formula for the running time T(n) of the algorithm

B. (3 points) Use the Master Formula to specify the complexity class to which T(n) belongs.

SCI(3Points)

Elaborate upon a parallel between points and topics in Algorithms and one or more SCI principles. This is a short essay; more credit will be awarded for richer content.