READ THE FOLLOWING INSTRUCTIONS CAREFULLY
BEFORE STARTING THE TEST
WHEN YOU ARE FINISHED, UPLOAD THIS EXAM TO CANVAS
WITH YOUR ANSWERS MARKED

COMP 3270 — Introduction to Algorithms
Spring 2020
Final Exam

EXAM IS OPEN TEXT AND NOTES
ALL ELECTRONIC DEVICES, INCLUDING CALCULATOR, SMARTPHONE, LAPTOP, TABLET OR SMARTPHONE ARE ALLOWED.

50 multiple choice questions
Each carries 2 points
20% course credit

Mark answers in this PDF file using "Highlight Text" feature of Adobe Acrobat Reader.
Upload the PDF file to Canvas **before the due date and time** to submit the exam.

Each question has exactly **ONE** correct answer
Different questions have different difficulty levels
Do not get stuck on any one question!

This test requires 150 minutes to complete.
But you have an **extended duration of 300** minutes to complete this test.

You should do your own work.
Any evidence of cheating will result in a zero grade and additional penalties/actions.

Submission deadline will be strictly enforced.
Late submissions will NOT be accepted.

Available at 2:00 PM Tuesday April 28
Due before 7:00 PM Tuesday April 28

Consider the problem of determining whether or not there exists at least one integer i such that A[i]=i in a non-empty array A[first,…,last] of distinct (no-duplicates) integers that is already sorted in the increasing order.

Strategy I: This problem of determining whether there exists an integer i such that A[i]=i in a non-empty sorted array A[first,…,last] of distinct integers can be solved by scanning the array from cell "first" to cell "last" checking if any cell contains an integer equal to that cell's index; if so return true else return false.

Strategy II:
1. If the array has only one cell, return true if the integer in it is the same as that cell's index; otherwise return false.
2. Otherwise determine the middle cell of A and return true if the integer in this cell is the same as that cell's index; otherwise:
    2.1 If the value in that cell is less than that cell's index, recursively solve the same problem for the left half of the array and return the resulting Boolean value.
    2.1 If the value in that cell is more than that cell's index, recursively solve the same problem for the right half of the array and return the resulting Boolean value.

1. Which of these statements is true?
    A. strategy I is less efficient than Strategy II
    B. strategy I more efficient than Strategy II
    C. strategy I is as efficient than Strategy II
    D. there is not enough information to compare the efficiencies of these two strategies
    [✓ E. none of these statements are true]

2. Which of these statements is true?
    A. strategy I is correct and strategy II is correct
    B. strategy I is incorrect and strategy II is correct
    [✓ C. strategy I is correct and strategy II is incorrect]
    D. strategy I is incorrect and strategy II is incorrect
    E. none of these statements are true

3. If four algorithms to solve the same problem have the following complexities: $O(n^3)$, $\Omega(n^3)$, $o(n^3)$, $\Theta(n^3)$, what is a reasonable assumption about which is the most efficient (in the absence of any additional information)?
    A. $O(n^3)$ algorithm   B. $\Omega(n^3)$ algorithm   [✓ C. $o(n^3)$ algorithm]   D. $\Theta(n^3)$ algorithm
    E. None of the above four algorithms is likely to be the most efficient

fib(n: non-negative integer) returns non-negative integer
1 if n==0 or 1 or 2 then return 1
2 else return fib(n-1)+fib(n-2)+fib(n-3)
Consider a variant of the Fibonacci algorithm shown above.

4. Is this algorithm tail-recursive?
    yes (mark T)   [✓ no (mark F)]

5. Does this algorithm duplicate work?
    [✓ yes (mark T)]   no (mark F)

6. If the original Fibonacci algorithm has the complexity of $O(2^n)$, what is a reasonable estimate of the complexity of the variant algorithm above?
A. $O(2^n)$ B. $O(n^2)$ [✓ C. $O(3^n)$]   D. $O(n^3)$ E. none of these

Mystery(n: non-negative integer) returns non-negative integer
```
if n ≤ 1 then return 0
else return (1+Mystery(n-2))
```

7. What is the mathematical function Mystery computes?

A. 2n B. 2^n [✓ C. floor(n/2)] D. ceiling(n/2) E. none of these

8. The precise meaning of algorithm correctness is:

A. A correct algorithm produces the correct outputs for all inputs.
B. A correct algorithm produces the correct output for any input.
C. A correct algorithm will not get into an infinite loop or infinite recursion for any input.
[✓ D. A correct algorithm produces the correct outputs for all valid inputs and halts.]
E. None of the above.

9. If you want to prove that an algorithm is correct, which of the following techniques can be used?

I. Proof by induction II. Proof by contradiction III. Proof using loop invariants IV. Proof by counterexample

A. III and IV only
B. I and II only
C. I, II and IV
[✓ D. I, II and III]
E. all of the above

10. [✓ True] or False? The following algorithm to move any and all zeroes in the array A to its right end is correct.

Move-Zeroes(A[p...r] of numbers, p≤r)
```
i = r + 1
for j = r down to p
    if A[j] ==0
        i = i – 1
        swap A[i] and A[j]
```

Find-Max(A:array[i…j] of numbers)
```
max=A[i]
for k=(i+1) to j
    if A[k]>max then max=A[k]
return max
```

11. What is an appropriate Loop Invariant to prove that this algorithm correctly finds the max number in the input array A[i...j]?

[✓ A. before any execution of the for loop with loop variable k having value p, (i+1)≤p≤j, max contains the first number in A]

B. before any execution of the for loop with loop variable k having value p, (i+1)≤p≤j, max contains the maximum value in the subarray A[i...(p–1)]

C. before any execution of the for loop with loop variable k having value p, (i+1)≤p≤j, max contains the maximum value in the subarray A[i...p]

D. before any execution of the for loop with loop variable k having value p, (i+1)≤p≤j, max contains the maximum value in the subarray A[i...(p+1)]

E. before any execution of the for loop with loop variable k having value p, (i+1)≤p≤j, max contains the maximum value in the subarray A[p...j]

Power-of-2(n: non-negative integer) returns non-negative integer

```
1    if n==0 then return 1
2    else return 2*Power-of-2(n–1)
```

Consider an inductive proof of correctness of the above algorithm and answer the following question.

12. What is the most correct statement of the **base case** proof?

A. when n=0 the algorithm returns 1 by step 1

B. when n=0 the algorithm returns the correct answer

C. when n=0 the algorithm does not make any recursive call

[✓ D. when n=0 the algorithm returns 1 by step 1, $2^0=1$, so the correct answer is returned.]

E. none of these is a correct base case proof

13. A recursive algorithm is characterized by the following recurrences:

$T(n)=2T(n/4)+T(3n/4)+cn$

$T(n)=c \qquad 0\leq n\leq 3$

I. This algorithm has 4 base cases

II. When input is not a base case, each execution spawns three recursive executions.

III. These recursive executions reduce input size by a factor of 4 and 1.3 respectively.

IV. In addition to spawning new recursive executions, the algorithm does a linear amount of work when the input is not a base case.

Which of the following is correct?

A. only one of the above four statements is true

B. only two of the above four statements are true

C. only three of the above four statements are true

D. none of the above four statements is true.

[✓ E. all of the above statements are true.]

$T(n) = T(n-1) + 15\lg n$; $T(1) = 1$

14. What is the first step of solving the recurrences above using the Forward Substitution method?

[✓ A. T(1)=1 so T(2) = T(1) + 15log2 = 16]

B. Make a guess of the solution and prove it inductively

C. T(n-1) = T(n-2) + 15lg(n-1) so T(n) = T(n-2) + 15(lg(n-1) + lgn)

D. Draw a Recursion Tree

E. Apply the Master Theorem

15. If you were to solve these recurrences using the Recursion Tree method, how many levels will the Recursion Tree have?

A. n+1 [✓ B. n] C. n-1 D. lgn E. none of these

16. A is an array of numbers sorted in the ascending order. If it were a Binary Heap, it would be a:

A. Max Heap [✓ B. Min Heap] C. Both Min and Max Heap D. It can't be a Heap

E. none of these

1	2	3	4	5
26	57	36	12	13

17. If Quick Sort with Median of Three Partitioning is called with the array above as input, what will this array look like after the very first Partition is completed in the original call to the algorithm?

A. [12, 13, 36, 26, 57] B. [26, 13, 12, 36, 57]

[✓ C. [13, 12, 26, 57, 36]] D. [13, 12, 36, 57, 26]

E. None of these

1	2	3	4	5	6
1_1	1_2	1_3	1_4	1_5	1_6

18. If the above shown array A is input to Counting Sort, where A contains six 1's (the subscripts merely indicate the relative order of these numbers, i.e., 1_1 is the first 1, 1_2 is the second 1, 1_3 is the third 1, and so on) and the range of inputs 0–k is 0–5, which of the following is the correct output (i.e., array B) produced by the algorithm (the subscripts indicate the original positions of these numbers in A)?
[✓ A. B = [1_6, 1_5, 1_4, 1_3, 1_2, 1_1]]
B. B = [1_1, 1_6, 1_2, 1_5, 1_3, 1_4]
C. B = [1_1, 1_2, 1_3, 1_6, 1_5, 1_4]
D. B = [1_1, 1_2, 1_3, 1_4, 1_5, 1_6]
E. B will contain six 1's but their order cannot be predicted

19. Why does Radix Sort need a stable sorting algorithm?
A. A stable sorting algorithm ensures that the time needed by Radix Sort is stable
B. A stable sorting algorithm ensures that the time needed by Radix Sort is independent of input size
C. A stable sorting algorithm ensures that the space needed by Radix Sort is stable
D. A stable sorting algorithm ensures that the space needed by Radix Sort is independent of input size
[✓ E. None of the above]

20. [✓ True] or False? If we limit the input numbers to Bucket Sort to be only integers on the range 0 to k where k is some integer greater than zero, and modify the algorithm to have k+1 buckets (a bucket each for integers 0,1,2,...k), i.e., in step 2 let B be an array [0...k] pointing to k+1 buckets, and we eliminate (i.e. delete) steps 7 and 8, it is guaranteed that for any input array A of integers in the range 0–k step 9 will produce the correct sorted output.

21. What is the complexity of Binary Search, which looks for a number X in an already sorted array by looking to see if the middle number is X, and if not, searching only the left or the right half of the array recursively depending on whether the middle number in the array is greater or less than X?
A. constant time [✓ B. Polylogarithmic] C. Polynomial D. Exponential E. Combinatorial

22. Insert 3,1,4,6,9,2,5,7 one after the other into an initially empty Binary Search Tree (BST) in the given order. After all the insertions, delete the root of the BST. What will be the root of the resulting BST after this deletion?
A. 1 B. 3 [✓ C. 4] D. 5 E. none of these

Naïve-String-Matcher(T,P: strings represented as arrays of character)

```
1 n=T.length
2 m=P.length
3 for s=0 to n-m
4         if P[1…m]==T[s+1…s+m]          //i.e., these subarrays match
5                 print "pattern occurs with shift" s
```

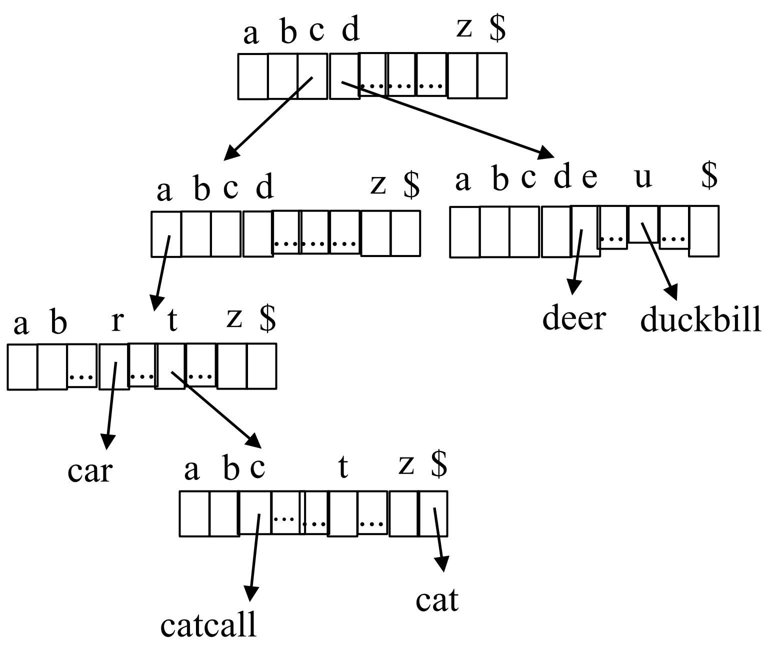
23. If T=abcarcarcadefn and P=carca, what are the values of shift s printed out by **Naïve-String-Matcher**?
A. only 2 B. only 3 [✓ C. 2 and 5] D. 3 and 6 E. none of these

24. If we want to extend the Rabin-Karp algorithm to the problem of searching a text string for occurrences of <u>a given set of k patterns</u> instead of just one pattern, where <u>all k patterns have the same length m</u>, how will the original algorithm have to be modified?
1. We need to initialize k variables $h_1 \ldots h_k$ in step 3.
2. We need to initialize k variables $p_1 \ldots p_k$ in step 4.

3. We need to calculate k numerical values for $p_1 \ldots p_k$ in step 7.
4. We need to compare t_s against each of these k numerical values and print appropriate messages in steps 10-12.
5. We need to calculate k numerical values $t_{s+1} \ldots t_{s+k}$ in step 14

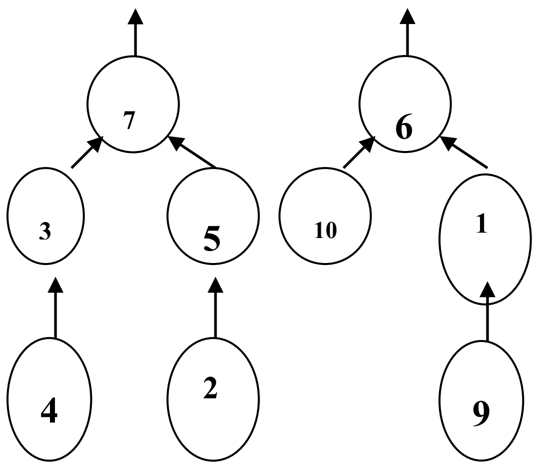
Two of the steps above are unnecessary and will in fact render the modified Rabin-Karp algorithm incorrect. Which ones are those?

A. 1 & 2 B. 3 & 5 C. 3 & 4 D. 2 & 4 [✓ E. 1 & 5]



25. A trie with five interior nodes (an interior node is a node containing an array of 27 pointers indexed by a,b,...,y,z,$) is shown above.
Statement I: If we delete "cat" from this Trie, the esulting Trie will still have 5 interior nodes.
Statement II: If after deleting "cat," we were to delete "catcall," the resulting Trie will have 4 interior nodes.
A. Statement I is true and statement II is false.
B. Statement I is false and statement II is true.
C. Statement I is false and statement II is false.
[✓ D. Statement I is true and statement II is true.]
E. We cannot say anything about the truth or these statements

26. Assuming union-by-size and path compression are used, what is the result of unioning the following two trees representing two Disjoint Sets and then executing a Find(9)?



A. There will be only one tree
B. The depth of the resulting tree will be 2
C. The root of the resulting tree will be 7
D. Nodes 1 and 9 will be connected directly to the root
[✓ E. All of these will happen]

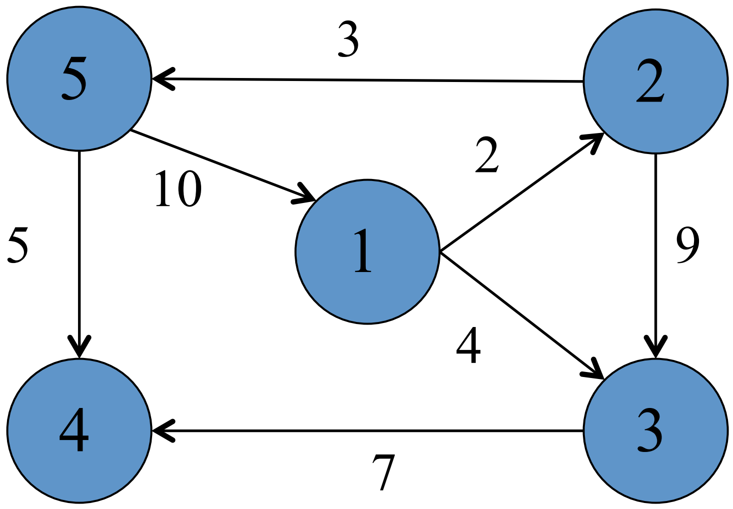
Suppose we want to add an operation Remove to the Disjoint Set data structure. It removes a data element e and all nodes below it (i.e. the subtree rooted at e) from the tree in which e is a node, and places them in a separate tree (thus making a new Disjoint Set), unless e is the root of a tree (If e is the root, then nothing is done). For example, "Remove 5" applied to the Disjoint Sets shown above will result in a new tree rooted at 5 with 2 as a child. A partial algorithm for this operation is given below. The disjoint sets are implemented using array P and unions are by size and find is done with path compression.

Remove(e, P)

```
1 if P[e] is not negative then
2         root = Find(e, P)
3         size = Size(e, P)          //Size(e, P) returns the size of the subtree rooted at e
4         P[e] = –size
5         _______________
```

27. What should step 5 be?
A. P[e] = –size
B. P[root] = P[root]+size
C. P[e] = size
[✓ D. P[root] = P[root]–size]
E. None of these

28. How many cells of the Adjacency Matrix representation of the graph below will have zeroes in them?

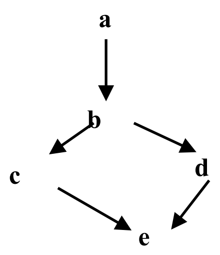


A. 25 B. 7 [✓ C. 18] D. 14 E. None of these

29. Given the Adjacency Matrix representation A of an undirected graph G, how can you turn it into the Adjacency Matrix A' of a graph G' that is the complement of G? The complement of a graph G is a graph G' with the same vertices such that two distinct vertices of G' are adjacent (i.e., connected by an edge) if and only if they are not adjacent in G.
A. By turning all 0's in A into 1's
B. By turning all 1's in A into 0's
[✓ C. By turning all 0's in A except the diagonal entries into 1's and all 1's in A into 0's]
D. By computing the transpose of A
E. none of the above

30. Consider a graph with n nodes and m edges. How many total array cells and linked list nodes will its Adjacency List have if the graph is undirected?
A. n cells and m nodes
[✓ B. n cells and 2m nodes]
C. 2n cells and m nodes
D. 2n cells and 2m nodes
E. none of the above

31. If Depth First Search (DFS) is done on the graph on the right side, starting with node "a", and if adjacent nodes are visited by the algorithm in alphabetical order, what is the order in which nodes are visited by DFS?
A. a b c d e B. a b d e c [✓ C. a b c e d] D. a b d c e
E. None of these



32. If Breadth First Search is done on the same graph starting with node "a", and if adjacent nodes are enqueued by the algorithm in alphabetical order, what is the order in which nodes are visited by BFS?
[✓ A. a b c d e] B. a b d e c C. a b c e d D. a b d c e
E. None of these

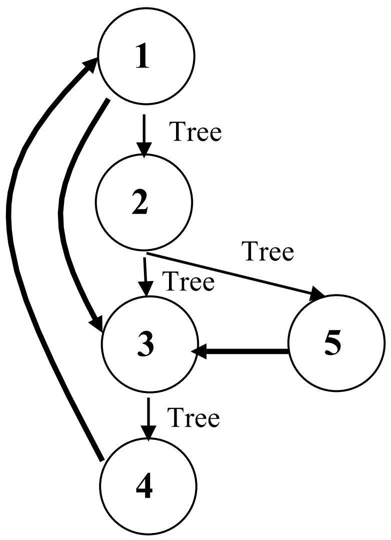
Assume that after running the Breadth First Search on some graph, the d and π values for each node are obtained as below.

Node	d	π
1	1	2
2	0	NIL
3	3	5
4	1	2
5	2	1

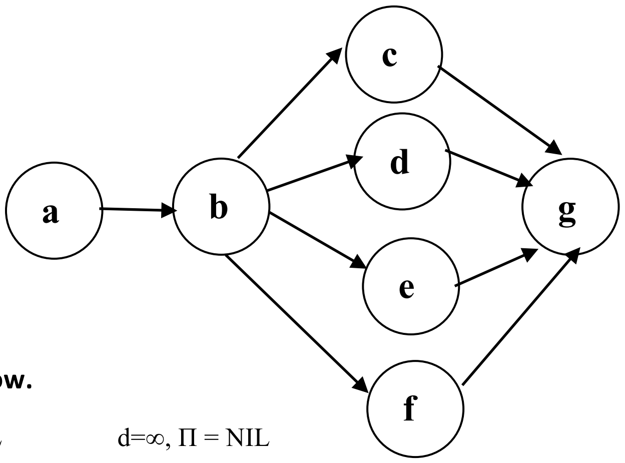
33. Which node was the starting node for the algorithm?
A. node 1 [✓ B. node 2] C. node 3 D. node 4 E. node 5

34. If the Depth-First Forest created by the DFS algorithm contains a tree with a backward edge, what does it mean?
A. The graph is directed.
B. The graph is connected.
C. The graph is biconnected.
[✓ D. The graph has a cycle.]
E. The graph is undirected.

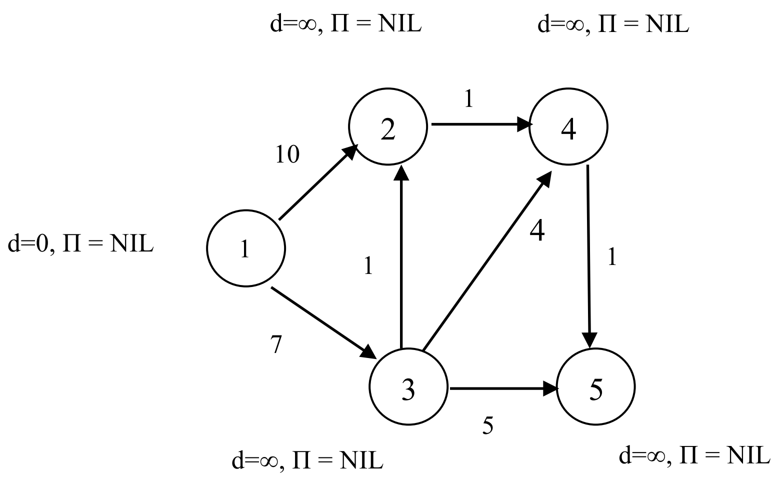
35. If the Depth-First Forest created by the DFS algorithm from a directed graph is as shown to the right (the forest has only one tree and its edges are marked), with the remaining edges of the graph also included in the picture, what are the classifications of the three unlabeled (in bold) edges (4->1),(1->3) and (5->3) respectively?
A. Forward, cross and back
[✓ B. Back, forward and cross]
C. Cross, back and forward
D. Forward, back and cross
E. Cross, forward and back



36. What is the topological sort of this graph?
    A. a b c d e f g
    B. a b d c e f g
    C. a b c d f e g
    D. a b f e d c g
    [✓ E. All of the above]



The next five questions refer to the graph below.



37. If Relaxation is applied to the edge from node 1 to node 2, what will be the values of d and Π for node 2 after the relaxation step?
A. 0, 1 B. 10, 0 [✓ C. 10, 1] D. 2, 1 E. None of these

38. Suppose Bellman-Ford algorithm is run on this graph with node 1 as the start node. The graph above shows the d and Π values of all nodes after step 1: INITIALIZE-SINGLE-SOURCE. True or [✓ False]? The algorithm will execute step 7 and return false.

39. What is the order in which the DAG-SHORTEST-PATHS algorithm, with node 1 as the start node, will consider nodes of the graph above in step 3?
A. 1,2,3,4,5 [✓ B. 1,3,2,4,5] C. 1,2,3,5,4 D. 1,3,2,5,4 E. None of these

40. What will be the values of d and Π for node 4 after the DAG-SHORTEST-PATHS algorithm, with node 1 as the start node, finishes executing on the graph above? The graph shows the d and Π values of all nodes after step 2: INITIALIZE-SINGLE-SOURCE.
[✓ A. 11, 2] B. 9, 2 C. 11, 3 D. 9, 3 E. None of these

41. If Dijkstra's algorithm is run on the graph above, what will be the values of d and Π for node 5 after the algorithm finishes? The graph shows the d and Π values of all nodes after step 1: INITIALIZE-SINGLE-SOURCE.
A. 10, 3 [✓ B. 12, 3] C. 10, 4 D. 12, 4 E. None of these

You are managing a large scale construction project with hundreds of subprojects, some which have to be completed before others can begin whereas some subprojects can be carried out simultaneously. The project and its subprojects, and the presence of dependencies between subprojects (which subprojects have to be done before which), can be modeled as a connected unweighted directed graph with nodes representing subprojects and directed edges representing dependencies.

42. Which of the following algorithms will allow you to determine if there is a cycle of dependencies among subprojects in this large project that will prevent you from completing the project?
A. Depth-first search
B. Any algorithm to find a Hamiltonian Circuit
C. Topological sort
D. Any algorithm to find a Vertex Cover
E. Any algorithm to find a Clique

43. Which of the following algorithms will allow you to determine a way to do the subprojects one after the other so that the construction project can be completed while ensuring that no subproject is started before all subprojects that have to be done before it are completed?
A. Depth-first search
B. Any algorithm to find a Hamiltonian Circuit
C. Topological sort
D. Any algorithm to find a Vertex Cover
E. Any algorithm to find a Clique

44. You have designed and built a wired computer network for the Pentagon. The top brass gives you a list of key commanders and asks you to check whether their computers are connected in such a way that even if any one commander's computer is taken down by a cyber attack, the computer of any other commander will be able to communicate with the computer of any other commander (except the one whose computer is down) over a direct network connection linking the two computers. Which algorithm can you use to check this?
A. Depth-first search
B. Any algorithm to find a Hamiltonian Circuit
C. Topological sort
D. Any algorithm to find a Vertex Cover
E. Any algorithm to find a Clique

45. The top brass then asks you to check whether there is a set of computers on the network such that if a cyber attack takes all of those down, then none of the remaining working computers will be able to communicate with any other working computer on the network. Which algorithm can you use to check this?
A. Depth-first search
B. Any algorithm to find a Hamiltonian Circuit
C. Topological sort
D. Any algorithm to find a Vertex Cover
E. Any algorithm to find a Clique

46. The top brass then asks you to verify whether all computers on the network are in working order once every day by sending a packet from your computer and routing it so that it goes to every other computer in the network and returns to your computer (if the packet returns to your computer you know all computers are working) with every other computer in the network receiving the packet and sending it to another computer only once. Which algorithm can you use to decide how to route the packet?
A. Depth-first search
B. Any algorithm to find a Hamiltonian Circuit

C. Topological sort
D. Any algorithm to find a Vertex Cover
E. Any algorithm to find a Clique

47. [✓ True] or False? If there is a problem that belongs to both class P and class NP-Complete, this implies that P=NP.

48. What is the distinguishing property of an NP problem?
A. It is known that a polynomial time algorithm to solve it can never be found.
[✓ B. No polynomial time or faster algorithm is currently known to solve it, but a polynomial time algorithm can check to see if a potential solution to it is correct or not.]
C. A polynomial time algorithm to solve it is known.
D. It has to do with digital circuits.
E. None of the above.

49. [✓ True] or False?
The logical proof of the undecidability (or uncomputability) of the halting problem (that no correct algorithm exists which takes any program P and any input I as its inputs and returns the answer "yes" if P will eventually halt after executing on I and "no" if P will never halt) can also be used to show that the problem of computationally determining whether any program P will eventually exhibit a particular behavior (such as "P will eventually print 'hello world' on the console") is also undecidable.

50. I learned something about algorithms from this course.
[✓ True] False