Comp 250: Final examination - VERSION #1

Instructors: Mathieu Blanchette and Jérôme Waldispühl

December 21st, 2015, 2:00 - 5:00pm

Examiner's signature:

Associate examiner's signature:

NAME:

STUDENT ID:

NOTE: DO NOT ANSWER ON THIS COPY.

- This is a multiple choices exam. For each question, only one answer can be provided.
- Answer the questions on the multiple choice page, using a lead pencil.
- You have 180 minutes to write the exam.
- This exam is worth 50% of your total mark.
- You can use up to 2 pages (1 double-sided sheet) of notes.
- Books and electronic devices are not allowed.
- If you believe that none of choices provided for a given question are correct, provide the answer that is the closest to being correct.
- This exam contains 45 questions on 20 pages.
- This examination is printed on both sides of the paper
- This examination paper must be returned
- The examination Security Monitor Program detects pairs of students with unusually similar answer patterns on multiple-choice exams. Data generated by this program can be used as admissible evidence, either to initiate or corroborate an investigation or a charge of cheating under Section 16 of the Code of Student Conduct and Disciplinary Procedures.

1. (4 points) Consider a sorted array of n distinct integers. What will be the complexity of the best algorithm to find the number of pairs of elements in the array whose sum is equal to 100?

   A. $\Theta(n)$

   B. $\Theta(n \log(n))$

   C. $\Theta(n^2)$

   D. $\Theta(n^2 \log(n))$

   E. $\Theta(100)$

2. (4 points) What will be the complexity of the following $count(n)$ algorithm:

Algorithm 1 Count(int n)

```
1: int count=0;
2: for int i = n; i > 0; i=i/2 do
3:     for int j = 0; j < i; j++ do
4:         count++;
5:     end for
6: end for
7: return count
```

   A. $\Theta(\log(n))$

   B. $\Theta(n)$

   C. $\Theta(n \log(n))$

   D. $\Theta((n \log(n))^2)$

   E. $\Theta(n^2)$

Trees in Java

In assignment 4, you wrote a program to calculate the derivative of an algebraic expression represented by an expression tree. The operations supported included "add", "mult", "minus", etc. Suppose we want to allow one more type of operations, called "square", where $square(f(x)) = (f(x))^2$. Recall that $\frac{d(f(x)^2)}{dx} = 2 \cdot f(x) \cdot \frac{df(x)}{dx}$. Which lines of code complete the portion of code that would need to be added to the differentiate method you implemented in order to handle this new type of operation?

```
class expressionTreeNode {
    private String value;
    private expressionTreeNode leftChild, rightChild, parent;

// constructor
  expressionTreeNode(String s, expressionTreeNode l,
                      expressionTreeNode r, expressionTreeNode p) {
        value = s;  leftChild = l;  rightChild = r;  parent = p;
    }

   expressionTreeNode getLeftChild() { return leftChild; }
   expressionTreeNode getRightChild() { return rightChild; }

expressionTreeNode differentiate() {
/* Same beginning as in your assignment solution */
...
       if ((getValue()).equals("square")) {
            expressionTreeNode l,r;
            l=new expressionTreeNode("2",null,null, null);
            // LINE #1
            // LINE #2
        }
}
```

3. (4 points) Which line correctly completes LINE #1?

   A. r= new expressionTreeNode("mult", getLeftChild().deepCopy(), getLeftChild().differentiate(),null);

   B. r= new expressionTreeNode("mult", getLeftChild().differentiate(), getLeftChild().differentiate(),null);

   C. r = new expressionTreeNode("mult", getLeftChild().deepCopy(), getLeftChild().deepcopy(),null);

   D. r = new expressionTreeNode("mult", getLeftChild().differentiate(), getRightChild().deepCopy(),null);

   E. r = new expressionTreeNode("mult", getLeftChild().differentiate(), getRightChild().differentiate(),null);

4. (4 points) Which line correctly completes LINE #2?

   A. return new expressionTreeNode("add", l, r, null);

   B. expressionTreeNode("square", l, r, null);

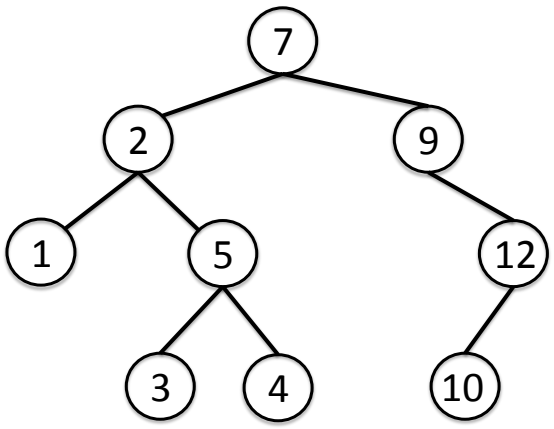
   C. return expressionTreeNode("add", l, r, null);

D. return expressionTreeNode("mult", l, r, null);

E. None of the above

Trees traversals

5. (4 points) Consider the following binary search tree:



What is the post order traversal for the given binary search tree ?

A. 7, 2, 9, 1, 5, 12, 3, 4, 10

B. 7, 2, 1, 5, 3, 4, 9, 12, 10

C. 1, 2, 3, 5, 4, 7, 9, 10, 12

D. 1, 3, 4, 5, 2, 10, 12, 9, 7

E. None of the above

6. (4 points) Consider the following pair of recursive algorithms calling each other to traverse a binary tree.

Algorithm 2 WeirdPreOrder(treeNode n)

```
1: if (n ≠ null) then
2:     print n.getValue()
3:     weirdPostOrder(n.getRightChild())
4:     weirdPostOrder(n.getLeftChild())
5: end if
```

Algorithm 3 WeirdPostOrder(treeNode n)

```
1: if (n ≠ null) then
2:     weirdPreOrder(n.getRightChild())
3:     weirdPreOrder(n.getLeftChild())
4:     print n.getValue()
5: end if
```

Which one of the following is the output being printed when weirdPreOrder(root) is executed on the binary tree shown above?

A. 7 12 10 9 5 4 3 1 2

B. 7 2 1 4 3 4 9 12 10

C. 1 2 3 5 4 7 9 10 12

D. 7 9 10 12 5 4 3 2 1

E. None of the above

Stacks

Consider the recursive algorithm below.

Algorithm 4 mistery(Stack stack)

```
1: value ← stack.pop()
2: if stack.isempty() then
3:     return value
4: else
5:     result ← mistery(stack)
6:     stack.push(value)
7:     return result
8: end if
```

We want to discover the purpose of this algorithm. First, we execute the following commands.

```
stack = New Stack();
stack.push("1");
stack.push("2");
stack.push("3");
print mistery(stack);
print mistery(stack);
print mistery(stack);
```

7. (4 points) What are the values printed during the execution of the instructions above.

   A. `3 2 1`

   B. `2 1 3`

   C. `1 2 1`

   D. `1 2 3`

   E. `3 3 3`

8. (4 points) What is this algorithm doing?

   A. It build an array-based representation of a heap with the numbers inserted in the stack.

   B. It simulates the removeMin() operation on a min heap.

   C. It removes and returns the element at the bottom of the stack.

   D. It sorts the numbers inserted in the stack.

E. It changes the content of the stack by reversing the order of its elements.

9. (4 points) If the stack contains n elements, what is the worst-case running time complexity of mistery()?

   A. $\Theta(1)$

   B. $\Theta(\log n)$

   C. $\Theta(n)$

   D. $\Theta(n \cdot \log n)$

   E. $\Theta(n^2)$

Binary Search Trees

We propose the following algorithm to insert new values in a binary search tree. Keys are integers.

Algorithm 5 rootinsert(Node root, int newvalue)

```
1:  newnode = new Node()
2:  newnode.setValue(newvalue)
3:  if newvalue ≤ root.getValue() then
4:      newnode.setRight(root)
5:      newnode.setLeft(root.left)
6:      root.setLeft(NULL)
7:  else
8:      newnode.setLeft(root)
9:      newnode.setRight(root.right)
10:     root.setRight(NULL)
11: end if
12: return newnode
```

10. (4 points) This algorithm is supposed to return a binary search tree in which we added a new key. Which of the following statements is correct?

    A. This algorithm is correct and runs in $\mathcal{O}(1)$.

    B. This algorithm is correct but produces trees with internal nodes having only one child.

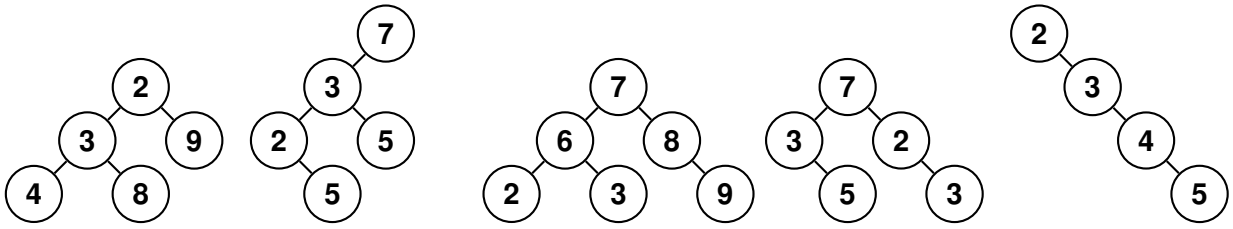
    C. This algorithm works if and only if the key that is being inserted has a value equal to the min or max value stored in the input binary search tree.

    D. This algorithm returns a valid binary search tree if there does not exist a key k such that $root.getValue() \leq k \leq newvalue$ or $newvalue \leq k \leq root.getValue()$.

    E. None of the the above.

Binary trees

Consider the following trees.



(A) (B) (C) (D) (E)

11. (4 points) Which of the tree above is a binary search tree? A. B. C. D. **E.**

12. (4 points) Which of the tree above is a heap? **A.** B. C. D. E.

Recursive algorithms

Consider the following recursive algorithm:

Algorithm 6 TestRecursive(int n)

Require: A non-negative integer n.

```
1: if (n = 0) then
2:     print "Zero"
3:     return
4: else
5:     TestRecursive( ⌊n/2⌋)
6: end if
7: return
```

13. (4 points) What will be printed when testRecursive(12) is executed? Assume that everything that gets printed appears on the same line.

    A. Zero 1 3 6 12

    B. 12 6 3 1 0 Zero

    C. 12 6 3 1 Zero

    D. Zero 0 1 3 6 12

    E. None of the above

14. (4 points) Suppose we are using the ProgramStack class seen in assignment 3 to use a stack to model recursion. What is the content of the ProgramStack at the point of the execution where "Zero" gets printed, during the execution of testRecursive(2)? Assume that the top of the stack is shown on the left.

A. [PC= 2, n=0], [PC= 5, n=1], [PC= 5, n=2]

B. [PC= 5, n=2], [PC= 5, n=1], [PC= 2, n=0]

C. [PC= 2, n=2], [PC= 1, n=1], [PC= 0, n=0]

D. [PC= 2, n=2], [PC= 5, n=2], [PC= 5, n=2]

E. None of the above

15. (4 points) Consider two functions $f(n)$ and $g(n)$. In a proof by induction to show that $\forall n \geq 1 : f(n) = g(n)$, what is the induction hypothesis?

A. $\forall k \geq 1 : \ f(k) = g(k)$

B. $f(x) = g(x)$ if $x = 1$

C. $f(x) = g(x)$ if $x = k$

D. $\forall n \geq k : \ f(n) = g(n)$

E. $f(n + 1) = g(n + 1)$

16. (4 points) Consider an undirected graph G whose adjacency matrix is A, i.e. $A[x, y] = 1$ if there is an edge between vertices x and y, and $A[x, y] = 0$ otherwise. If $B = A^2$, what is the value of $B[x, y]$?

A. 1 if there is a path of length 2 from x to y, and 0 otherwise

B. 1 if there is a cycle of length 2 between x to y, and 0 otherwise

C. The number of paths of length 2 from x to y

D. The number of cycles of length 2 between x and y

E. None of the above

Short questions

17. (4 points) What should one do in order to increase the page-rank of his web site W, as defined in class?

A. Add links from W to several "high-profile" pages

B. Reduce the number of links going from W to other sites

C. Ask professors to put a link to W from their own web sites

D. Ask friends who already have a link to W to also include links to "high-profile" pages

E. None of the above

18. (4 points) Consider a country where coins have value 1, 5, and 6. On which of the following totals will the greedy algorithm fail to obtain an optimal solution?

A. 8

B. 9

C. 10

D. 11

E. 12

19. (4 points) Consider the 5-queen problem seen in class, which consists of placing 5 queens on a 5x5 chessboard so that no two queens attack each other. Suppose that the backtracking algorithm seen in class is used to find a solution, and that halfway through its execution, its current partial solution is shown below. When the algorithm encounters its first valid solution, which of the squares on the last row will be occupied by a queen?

	*			
				*
*				
A	*B*	*C*	*D*	*E*

A. Square A

B. Square B

C. Square C

D. Square D

E. Square E

20. (4 points) When writing Java programs, which of the following statements is/are true?
1. One should always prefer variables of type "double" rather "float", because "double" is more precise.
2. Because they return nothing, methods that have return type "void" should be avoided.
3. A constructor method can only be executed at the beginning of the execution of the program.
4. A program should be broken up in as many methods as possible.

A. 1 and 2 only

B. 1, 2, and 3 only

C. 1, 2, and 4 only

D. 2, 3, and 4 only

E. None of the above

21. (4 points) What is/are the main advantage(s) of public-key cryptography over secret-key cryptography?

A. The sender (Alice) and receiver (Bob) do not have to secretely agree on an encryption key before being able to exchange message.

B. Public-key cryptography is safer

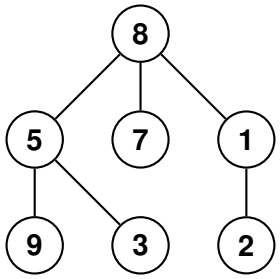
C. Public-key cryptography is faster

D. Message encrypted with public-key cryptography are more compact

E. All of the above

Array-based representation of trees

A ternary tree is a tree such that each internal node has at most 3 children. As illustrated below, ternary trees can be represented with an array.



Graphical representation

0	1	2	3	4	5	6	7	8	9	10	11	12	13
-	8	5	7	1	-	9	3	-	-	-	-	2	-

Array-based representation

22. (4 points) Let i be the index of a node n. We want to exchange the left child of n with the right child of n. What are the indices of the nodes we want to exchange?

A. $i \leftrightarrow 2 \cdot i + 2$

B. $i \leftrightarrow 3 \cdot i - 1$

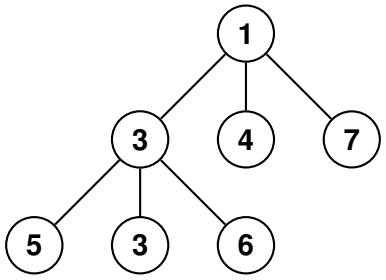
C. $3 \cdot i - 1 \leftrightarrow 3 \cdot i + 1$

D. $2 \cdot i \leftrightarrow 3 \cdot i + 1$

E. $2 \cdot i \leftrightarrow 2 \cdot i + 1$

Heaps

We can implement ternary heaps just as like the binary heaps. We show in the following figure an example of a min ternary heap.



23. (4 points) What **advantage** does a ternary heap have versus a binary heap?

A. We can store more keys.

B. For the same number of keys, the height of the tree will be smaller.

C. The array-based representation is more compact.

D. In heapify and delete operations, only one comparison is needed to get the minimum of 3 children while swapping for maintaining heap property.

E. None.

24. (4 points) What **disadvantage** does a ternary heap have versus a binary heap?

A. We cannot store has many keys as with a binary tree.

B. For the same number of keys, the ternary tree will be higher.

C. It cannot be represented with an array.

D. In heapify and delete operations, it will take 2 comparisons instead of one to get the minimum of 3 children while swapping for maintaining heap property.

E. None.

Graph problems

Which classical graph problem can be used to appropriately models the following situations?

25. (2 points) A taxi driver is looking for the fastest possible route to take his client to her destination.

A. Eulerian Cycle

B. Hamiltonian Cycle

C. Maximum Clique

D. Graph coloring

E. None of the above

26. (2 points) A police car needs to patrol every street in the city exactly once during their shift.

A. Eulerian Cycle

B. Hamiltonian Cycle

C. Maximum Clique

D. Graph coloring

E. None of the above

27. (2 points) A set of n tasks need to be performed. Several tasks can be performed in parallel, except for certain pairs of tasks that are conflicting and that cannot be performed together. We want to schedule all the tasks to the smallest possible number of time slots.

A. Eulerian Cycle

B. Hamiltonian Cycle

C. Maximum Clique

D. Graph coloring

E. None of the above

Analysis of recursive algorithms

28. (4 points) You have discovered a new algorithm that takes two parameters, m and n, where $m \leq n$. Upon some analysis, you find that the runtime obeys the following formula:

$$T(m,n) = \begin{cases} 2 \cdot T(m-1, n-1) & \text{if } m \geq 1 \\ T(0, n-1) + 1 & \text{if } m = 0 \text{ and } n \geq 1 \\ 1 & \text{if } m = 0 \text{ and } n = 0 \end{cases}$$

Give an explicit formula for $T(m,n)$.

A. $T(m,n) = 2^{n+1} \cdot 2^m$

B. $T(m,n) = 2^m$

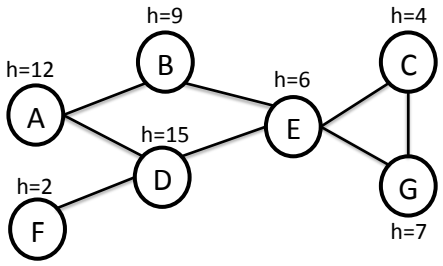
C. $T(m,n) = 2^{m-n+1}$

D. $T(m,n) = 2^m \cdot (n-m+1)$

E. None of the above

Graphs

Consider the following undirected graph (ignore the values of the h variables for now).



In class, we saw two graph traversal algorithms: depth-first search (DFS) and breadth-first search (BFS). Their pseudocode is given below.

Algorithm 7 DFS(Graph G, vertex v

```
1:  print v
2:  v.setLabel(VISITED)
3:  for all u in v.getNeighbors() do
4:      if ( u.getLabel() ≠ VISITED ) then
5:          DFS(G, u)
6:      end if
7:  end for
```

Algorithm 8 BFS(Graph G, vertex v

```
1:  q = new Queue()
2:  v.setLabel(VISITED)
3:  q.enqueue(v)
4:  while (! q.empty()) do
5:      w ← q.dequeue()
6:      print w
7:      for all u in w.getNeighbors() do
8:          if (u.getLabel() ≠ VISITED) then
9:              u.setLabel(VISITED)
10:             q.enqueue(u)
11:         end if
12:     end for
13: end while
```

29. (4 points) What will be printed when DFS(Graph, A) is executed (i.e. a depth-first search starting at node A of the above graph)? Assume that v.getNeighbors() method returns the neighbors of v in alphabetical order.

    A. A B C D E F G

    B. A B D E F C G

    C. A B E C D G F

    D. A B E C D F G

    E. None of these answers

30. (4 points) What will be printed by iterativeBFS(Graph, A) (i.e. a breadth-first search starting at node A of the above graph)? Again, assume that v.getNeighbors() method returns the neighbors of v in alphabetical order.

    A. A B D E F C G

    B. A D B F E G C

    C. A B E C G D F

    D. A B D E C G F

    E. None of these answers

Now, assume that the graph represents the roadmap of a city, with edges representing streets and nodes representing intersections. Each node has a given altitude (h), as shown on the graph. Suppose a major water leak happens at a given intersection. From that node, water flows along the streets (i.e. the edges incident onto that node), but only if the street is

downhill (i.e. water flows from node u to node v if and only if (i) node u is flooded and (ii) altitude(u)>altitude(v)). For example, if the leak happens at node A, the water will flow to node B but not D. From B, water will flow to E. From E, water will flow to C but not to G. Water will never reach nodes D, F, and G. We say that nodes A, B, and E undergo a "flowing flood", because they are flooded but water is flowing elsewhere from them. We say that nodes C is under a "standing water flood" because it is flooded but from it water is flowing nowhere.

Complete the algorithm below, which prints out the set of vertices that will be flooded by a leak starting at vertex v, together with their flood status (flowing or standing water). The algorithm should only explore the portion of the graph that gets flooded. For example, when called on vertex A of the graph shown above, it should print (in whichever order you want):
A: flowing flood
B: flowing flood
E: flowing flood
G: standing water flood

Algorithm 9 assessFlood(graph G, vertex v)

```
1:  Input: A vertex v that is flooded in a graph G
2:  Output: Prints the flood status of all nodes that will be affected by the flood.
3:  q ← new MyADT() /* See question below */
4:  v.setLabel(VISITED)
5:  q.insert (v)
6:  while (! q.empty()) do
7:     w ← q.remove()
8:     isFlowing ← False;
9:     for all u in w.getNeighbors() do
10:       if ( ... LINE1...) then
11:          isFlowing ← true
12:          if ( u.getLabel() ≠ VISITED ) then
13:             u.setLabel(VISITED)
14:             q.insert(u)
15:             /* location A*/
16:          end if
17:          /* location B*/
18:       end if
19:       /* location C*/
20:    end for
21:    /* location D*/
22: end while
```

31. (4 points) In the pseudocode above, we have used the type MyADT. Which type(s) of ADT(s) would be required to make the algorithm work (assuming that insert()/remove() and replaced by their appropriate names for that ADT, e.g. push/pop or enqueue/dequeue)?

A. Queue only

B. Stack only

C. Priority queue only

D. Either a Queue or a Stack

E. None of the above

32. (4 points) Which of the lines below is needed at LINE1 of the algorithm?

A. getAltitude(u) < getAltitude(w)

B. getAltitude(u) > getAltitude(w)

C. getAltitude(u) < getAltitude(w) AND u.getLabel() ≠ VISITED

D. getAltitude(u) > getAltitude(w) AND u.getLabel() ≠ VISITED

E. None of the above

33. (4 points) What is the appropriate location for the following lines of code?

if (isFlowing) **then** print w+": flowing flood"
else print w+" standing water flood"

A. Location A

B. Location B

C. Location C

D. Location D

E. None of the above.

Recursive algorithms

34. (4 points) Consider the following *mystery()* recursive algorithm:

Algorithm 10 mystery(int a, int b)

Require: Two non-negative integers a and b.

```
1: if (b = 0) then
2:     return 0
3: else if (b mod 2 = 0) then
4:     return mystery( ____ , ____ )
5: else
6:     return mystery( ____ , ____ ) + a
7: end if
```

Examples of the recursive algorithm's behaviour are below:

input	$\Longrightarrow$	*output*
$mystery(2,\ 50)$	$\Longrightarrow$	100
$mystery(3,\ 35)$	$\Longrightarrow$	105
$mystery(2,\ 25)$	$\Longrightarrow$	50
$mystery(0,\ 9)$	$\Longrightarrow$	0
$mystery(5,\ 5)$	$\Longrightarrow$	25

What arguments should replace the blank statments (i.e., ____) when recursively calling the *mystery()* algorithm (**lines 4 and 6**) to produce the above output?

A. 0 and $b - a$

B. $a + b$ and b / 2

C. a / a and b / b

D. a / 2 and $b + b$

E. $a + a$ **and** b / 2

Design of recursive algorithms

35. (4 points) A palindrome is a string that reads the same in the forward and reverse directions, such as "laval" or "kayak". Complete the following recursive algorithm that returns `True` if the input string is a palindrome, and `False` otherwise. Given a string s, the substring from positions i to j (inclusively) is written as $s[i:j]$. A character at position i can be accessed by $s[i]$ and the index starts at 0.

Algorithm 11 PalindromeRecursive(String s)

Require: A string s.
1: **if** (length$(s) \leq 1$) **then**
2: **return** True
3: **else if** $s[0] = s[\text{length}(s) - 1]$ **then**
4: **return** PalindromeRecursive$(\cdots)$
5: **end if**

A. $s[0 : \text{length}(s)]$

B. $s[1 : \text{length}(s)]$

C. $s[0 : \text{length}(s) - 1]$

D. $s[1 : \text{length}(s) - 1]$

E. None of the above.

Recursive algorithms

Consider the "Tribonacci" sequence $T_0, T_1, T_2, ...$ defined as follows:

$$T_n = \begin{cases} n & \text{if } n \leq 2 \\ T_{n-3} + T_{n-2} + T_{n-1} & \text{if } n \geq 3 \end{cases}$$

36. (4 points) What is the value of T_6 ?

A. 6

B. 11

C. 20

D. 37

E. None of the above

37. (4 points) The following recursive algorithm computes T_n:

Algorithm 12 Trib(int n)

1: **if** (n < 3) **then**
2: **return** n
3: **else**
4: **return** Trib(n-3) + Trib(n-2) + Trib(n-1)
5: **end if**

Let $X(n)$ be the total number of times the Trib() method will be called during the execution of Trib(n) (including the call to Trib(n) itself). For example, $X(0) = 1, X(3) = 4, X(4) = 7$.

Write a recurrence for $X(n)$.

A. $X(n) = \begin{cases} 1 & \text{if } n < 3 \\ X(n-1) + X(n-2) + X(n-3) & \text{if } n \geq 3 \end{cases}$

B. $X(n) = \begin{cases} n & \text{if } n < 3 \\ X(n-1) + X(n-2) + X(n-3) & \text{if } n \geq 3 \end{cases}$

C. $X(n) = \begin{cases} 1 & \text{if } n < 3 \\ X(n-1) * X(n-2) * X(n-3) & \text{if } n \geq 3 \end{cases}$

D. $X(n) = \begin{cases} n & \text{if } n < 3 \\ X(n-1) * X(n-2) * X(n-3) & \text{if } n \geq 3 \end{cases}$

E. None of the above

Sorting

Suppose that while your computer is sorting an array of integers, its memory is struck by a cosmic ray that changes exactly one of the keys to something completely different.

38. (4 points) What is the **worst-case** possibility if you are using **MergeSort**?

A. The final array will be fully sorted except exactly one key.

B. The final array will have just one or two keys out of place.

C. The final array will consist of two separate sorted subsets, one following the other, plus perhaps one or two additional keys out of place.

D. The final array will not even be close to sorted.

E. The algorithm will stop and the final array will be incomplete.

MergeSort algorithm

The following algorithms are variants of the MergeSort algorithm. For each variant, indicate the statement that best describes it (note: define n = stop - start + 1).

39. (4 points) .

Algorithm 13 MergeSort1(int A[], int start, int stop)

```
1: if (start<stop) then
2:     mid ← (start+stop)/2
3:     merge(A, start, mid, stop)
4:     mergeSort1(A,start,mid)
5:     mergeSort1(A, mid+1, stop)
6: end if
```

A. The algorithm will always produce the correct answer and runs in time O(n log(n))

B. The algorithm will always produce the correct answer but does not runs in time O(n log(n))

C. The algorithm will not always produce the correct answer and runs in time O(n log(n))

D. The algorithm will not always produce the correct answer and does not runs in time O(n log(n))

E. The algorithm produce an infinite recursion

40. (4 points) .

Algorithm 14 MergeSort2(int A[], int start, int stop)

```
1: if (start<stop) then
2:     mid ← ⌊(start + stop)/2⌋
3:     MergeSort2(A,start,mid)
4:     MergeSort2(A, mid+1, stop)
5:     MergeSort2(A,start,mid)
6:     merge(A, start, mid, stop)
7: end if
```

A. The algorithm will always produce the correct answer and runs in time O(n log(n))

B. The algorithm will always produce the correct answer but does not runs in time O(n log(n))

C. The algorithm will not always produce the correct answer and runs in time O(n log(n))

D. The algorithm will not always produce the correct answer and does not runs in time O(n log(n))

E. The algorithm produce an infinite recursion

41. (4 points) .

Algorithm 15 MergeSort3(int A[], int start, int stop)

```
1: if (start<stop) then
2:     mid ← start
3:     MergeSort3(A,start,mid)
4:     MergeSort3(A, mid+1, stop)
5:     merge(A, start, mid, stop)
6: end if
```

A. The algorithm will always produce the correct answer and runs in time O(n log(n))

B. The algorithm will always produce the correct answer but does not runs in time O(n log(n))

C. The algorithm will not always produce the correct answer and runs in time O(n log(n))

D. The algorithm will not always produce the correct answer and does not runs in time O(n log(n))

E. The algorithm produce an infinite recursion

Big-O notation

42. (4 points) Consider the function $f(n) = \log(n!)$ and $g(n) = n\log(n^2)$. Which of the following statements holds?

    A. $f(n)$ **is** $O(g(n))$ **and** $g(n)$ **is** $O(f(n))$

    B. $f(n)$ is $O(g(n))$ but $g(n)$ is not $O(f(n))$

    C. $f(n)$ is not $O(g(n))$ but $g(n)$ is $O(f(n))$

    D. $f(n)$ is not $O(g(n))$ and $g(n)$ is not $O(f(n))$

    E. None of the above

43. (4 points) Consider the function $f(n) = n\log(n) + n + 1$ and $g(n) = 2n$. Which of the following statements holds?

    A. $f(n)$ is $O(g(n))$ and $g(n)$ is $O(f(n))$

    B. $f(n)$ is $O(g(n))$ but $g(n)$ is not $O(f(n))$

    C. $f(n)$ **is not** $O(g(n))$ **but** $g(n)$ **is** $O(f(n))$

    D. $f(n)$ is not $O(g(n))$ and $g(n)$ is not $O(f(n))$

    E. None of the above

44. (4 points) Consider two non-negative functions $f(n)$ and $g(n)$ such that $f(n)$ is $O(g(n))$. Which of the following statements may **not** hold?

    A. $\log(f(n))$ is $O(\log(g(n)))$

    B. $\log(f(n))$ is $O(g(n))$

    C. $f(n)$ is $O(2^{g(n)})$

    D. $\log(f(n))$ is $O(2^{g(n)})$

    E. $2^{f(n)}$ **is** $O(2^{g(n)})$

45. (4 points) What is the Big O notation for the best case and worst case running time of the SelectionSort algorithm seen in class?

    A. best case: $O(1)$, worst case: $O(n)$

    B. best case: $O(1)$, worst case: $O(n\log(n))$

    C. best case: $O(n)$, worst case: $O(n\log(n))$

    D. best case: $O(n\log(n))$, worst case: $O(n^2)$

    E. best case: $O(n^2)$**, worst case:** $O(n^2)$