1

Illinois Institute of Technology
Computer Science

Final Exam

CS 331 — Data Structures
Fall 2014
Wednesday, December 10, 2014 10:30–12:30

This is a **closed book** and **closed notes** exam.
You are **not** allowed to use calculators or computers during this exam.
Do **ALL** problems in this booklet. Read each question very carefully.
You may detach pages, but **you must return all pages of this exam.**
Your exam will be digitized for grading. **The back sides of pages will be considered scratch paper, and will be ignored for grading.** If you need extra space to write an answer, "overflow" sheets are provided at the end of the exam.

Name	
IIT Email	

Multiple Choice

Each question has exactly one correct answer. **You are allowed to select more than one answer.** You get one point for showing up to the exam, three points for circling the correct answer, and lose one point for every incorrect answer you circle. Thus, leaving a question blank will score one point. Circling the correct answer and an incorrect answer scores three points. Circling three incorrect answers scores negative two points. Thus, you can get partial credit, but you will be penalized for guessing.

If the idea of negative points scares you, circle only one answer for each problem and it will score exactly like a traditional multiple choice exam.

Select your choice by circling the corresponding letter. If you make a mistake, draw an "X" through the choice. If you really mess it up, cross them all out and draw a box clearly labeled with your answer. In the event that your answer is hard to read, all reasonable interpretations will be used. For example, a letter that looks like both an 'a' and a 'd' will be considered both. So be neat.

Question 1)$_{1260}$ (4 points)
What is an abstract data type?

a) a class that has some methods undefined, and cannot be instantiated directly.

b) a type in which the implementation is hidden, and access given via a public interface.

c) a class that has private methods.

d) a type that implements virtual memory rather than physical memory.

Question 2)$_{1261}$ (4 points)
Which of the following is **not** a benefit of abstract data types?

a) ensure the integrity of the representation of the data

b) enables users to reason about the concept rather than the implementation

c) causes the code to be self-documenting

d) can change implementation without breaking programs that use it

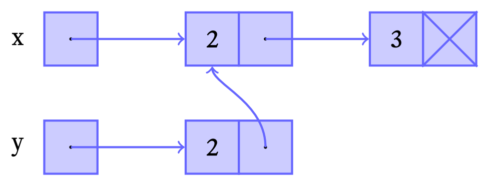
Question 3)$_{12c7}$ (4 points)

Suppose we have this Clojure code.

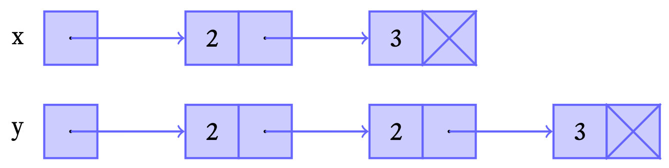
```
(def x (cons 2 (cons 3 nil)))
(def y (cons 2 x))
```

Which memory diagram best describes the result?

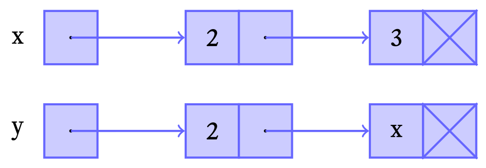
a)



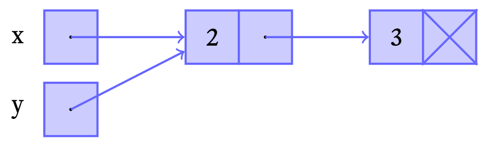
b)



c)

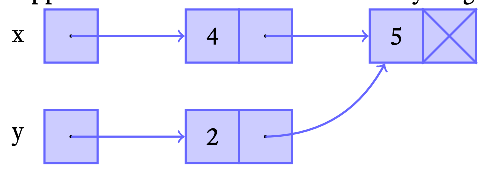


d)



Question 4)$_{12c8}$ (4 points)

Suppose we want to create this memory diagram.



We want to do this with vectors. Which code sequence can do this?

a)
```
(def x [4 5 2])
(def y [2 x])
```

b)
```
(def x [4 [5 nil]])
(def y [2 (x 1)])
```

c)
```
(def x [4 [5 nil]])
(def y [2 x])
```

d)
```
(def x [4 [5 nil]])
(def y (x 1))
```

Question 5)$_{12b3}$ (4 points)
If you have a mutable singly linked list, with no last pointer, here is the most efficient place to insert a new element?

a) the beginning

b) the end

c) both beginning and end

d) all inserts are the same

Question 6)$_{12b2}$ (4 points)
If you have a mutable singly linked list with a `last` pointer, which operation can **not** be done more quickly than a mutable singly linked list without a `last` pointer?

a) delete last element

b) get last element

c) insert before last element

d) insert after last element

Question 7)$_{127d}$ (4 points)
The "delete by relinking" method has trouble if a certain kind of element needs to be deleted. Which one is it?

a) the first element

b) the last element

c) an element that is duplicated

d) an element that is shared

Question 8)$_{1267}$ (4 points)
What is the time complexity of reversing a linked list?

a) $\mathcal{O}(1)$

b) $\mathcal{O}(\lg n)$

c) amortized $\mathcal{O}(1)$

d) $\mathcal{O}(n)$

Question 9)$_{12b4}$ (4 points)
Suppose I need a list, and I do not know how large it will be in advance, but once it's populated I won't have to change it much. What implementation is best?

a) persistent array list

b) persistent singly linked list

c) mutable doubly linked list

d) mutable array list

Question 10)$_{1269}$ (4 points)
If we already have a pointer to a doubly linked list node, what is the time complexity of removing it from a mutable doubly linked list?

a) $\mathcal{O}(1)$

b) $\mathcal{O}(\lg n)$

c) $\mathcal{O}(n)$

d) $\mathcal{O}(n^2)$

Question 11)$_{126a}$ (4 points)
What is the time complexity for deleting an element from a persistent doubly linked list?

a) $\mathcal{O}(1)$

b) $\mathcal{O}(\lg n)$

c) $\mathcal{O}(n)$

d) $\mathcal{O}(n^2)$

Question 12)$_{126b}$ (4 points)
What data structure is a FIFO?

a) array list

b) linked list

c) stack

d) queue

Question 13)$_{126c}$ (4 points)
What is the time complexity for `dequeue` for a standard queue?

a) $\mathcal{O}(1)$

b) $\mathcal{O}(\lg n)$

c) amortized $\mathcal{O}(1)$

d) $\mathcal{O}(n)$

Question 14)$_{126d}$ (4 points)
What is the time complexity for `dequeue` for an two-list queue?

a) $\mathcal{O}(1)$

b) $\mathcal{O}(\lg n)$

c) amortized $\mathcal{O}(1)$

d) $\mathcal{O}(n)$

Question 15)$_{126e}$ (4 points)
What is the time complexity for `push` for a standard stack?

a) $\mathcal{O}(1)$

b) $\mathcal{O}(\lg n)$

c) amortized $\mathcal{O}(1)$

d) $\mathcal{O}(n)$

Question 16)$_{126f}$ (4 points)
Which of the following is **not** a good implementation for a queue?

a) a persistent linked list

b) two persistent linked lists

c) a mutable linked list

d) an array/vector

Question 17)$_{1270}$ (4 points)
If you want to use an array/vector to implement a stack, what do you need?

a) a stack pointer to point to the first available space

b) a stack pointer to point to the last element inserted

c) a stack pointer to point to the first element inserted

d) a secondary vector to handle overflow

Question 18)$_{1271}$ (4 points)
What is the purpose of a sentinel?

a) eliminate boundary conditions

b) guard against overflow

c) makes doubly linked lists work in persistent environments

d) makes binary search trees work in persistent environments

Question 19)$_{1272}$ (4 points) What is the expected time complexity for inserting something into a binary search tree?

a) $\mathcal{O}(1)$

b) $\mathcal{O}(\lg n)$

c) $\mathcal{O}(n)$

d) $\mathcal{O}(n \lg n)$

Question 20)$_{1273}$ (4 points)
What is the worst case time complexity for inserting something into a binary search tree?

a) $\mathcal{O}(1)$

b) $\mathcal{O}(\lg n)$

c) $\mathcal{O}(n)$

d) $\mathcal{O}(n \lg n)$

Question 21)$_{12b6}$ (4 points)
What is a sequence of insertions that cause worst-case behavior for binary search trees?

a) 1,4,2,6,3,5,7

b) 4,2,6,1,3,5,7

c) 4,6,2,7,5,3,1

d) 1,7,2,6,3,5,4

Question 22)$_{1275}$ (4 points)
If you delete a node from a BST, and the node has one child, what should you do?

a) replace the node with `nil`

b) replace the node with the child

c) replace the node with an inorder successor or predecessor

d) replace the node with the parent

Question 23)$_{1276}$ (4 points)
If you delete a node from a BST, and the node has two children, what should you do?

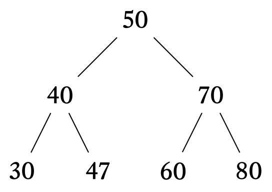
a) replace the node with `nil`

b) replace the node with the child

c) replace the node with an inorder successor or predecessor

d) replace the node with the parent

Question 24)$_{1277}$ (4 points)
Consider the following tree:



Which node is the inorder successor to 50?

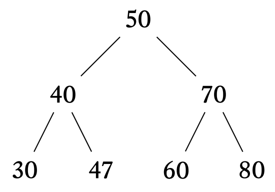
a) 47

b) 60

c) 70

d) 80

Question 25)$_{12c9}$ (4 points)
Consider the following tree:



If we insert a 49, where will it go?

a) as a parent of 47

b) as a left child of 47

c) as a right child of 47

d) as a left child of 30

Question 26)$_{1280}$ (4 points)
What do lists have that sequences do not have in CLOJURE?

a) A specific implementation

b) a `next` operation

c) a `first` operation

d) an ordering to the data

Question 27)$_{1295}$ (4 points)
The following code calls `rest` on a CLOJURE vector. What will be the result?

```
    (rest [2 4 6 8])
```

a) An exception

b) A sequence `(4 6 8)`

c) A vector `[4 6 8]`

d) A list `(4 6 8)`

Question 28)$_{1282}$ (4 points)
What advantages is there to using sequences?

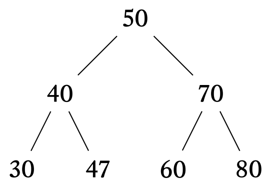
a) they take less memory

b) they are more flexible

c) they are more efficient

d) they are more accurate

Question 29)$_{12ca}$ (4 points) Consider the following tree:



Which of the following is a in-order traversal?

a) 30,40,47,50,60,70,80

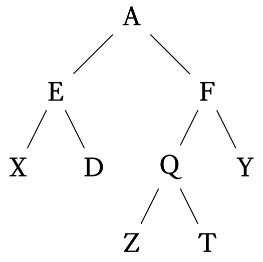
b) 30,47,40,60,80,70,50

c) 50,40,70,30,47,60,80

d) 50,40,30,47,70,60,80

Question 30)$_{12cb}$ (4 points)

Consider the following tree:



Note that this tree is not a BST. Which of the following is a preorder traversal?

a) A,E,F,X,D,Q,Y,Z,T

b) A,E,X,D,F,Q,Z,T,Y

c) X,E,D,A,Z,Q,T,F,Y

d) X,D,E,Z,T,Q,Y,F,A

Question 31)$_{1286}$ (4 points)

Which of the following is an advantage of depth-first search vs. breadth-first search?

a) It returns the solution closest to the root.

b) It takes less memory.

c) It is always faster.

d) It handles deeper data structures.

Question 32)$_{1285}$ (4 points)

Which of the following is an advantage of breadth-first search vs. depth-first search?

a) It returns the solution closest to the root.

b) It takes less memory.

c) It is always faster.

d) It handles wider data structures.

Question 33)$_{1289}$ (4 points)
Suppose you just read an integer from memory location 1000. According to the principle of spatial locality, which of the following can we expect?

a) The content of location 1000 will be temporary.

b) We will access the location 1001 soon.

c) We will need a local copy of location 1000.

d) We will access memory location 1000 again soon.

Question 34)$_{1299}$ (4 points)
The move-to-front heuristic is...

a) Fast and stable

b) Fast and unstable

c) Slow and stable

d) Slow and unstable

Question 35)$_{129a}$ (4 points)
Suppose we have a list with content `(1 2 4 8 16)`. We access element 4. What will be the result of the move-to-front heuristic?

a) `(1 2 8 4 16)`

b) `(4 1 2 8 16)`

c) `(1 2 4 8 16)`

d) `(4 8 16 1 2)`

Question 36)$_{12bc}$ (4 points)
Suppose we have the vector `[4 2 6 8 3 5 7]`. What will be the first three elements inserted if we run insertion sort on it?

a) 4,2,6

b) 2,3,4

c) 3,5,7

d) 5,6,7

Question 37)$_{12bb}$ (4 points)
Suppose we have the vector `[4 2 6 8 3 5 7]`. What will be the first three elements selected if we run selection sort on it?

a) 4,2,6

b) 2,3,4

c) 3,5,7

d) 5,6,7

Question 38)$_{128f}$ (4 points)
Which of the following is a special ability of selection sort?

a) It can sort most arrays in $\mathcal{O}(n)$ time.

b) It can give useful results if interrupted before it has finished.

c) It can return the elements closest to the beginning of the vector.

d) It runs in $\mathcal{O}(n)$ time if the data is already sorted.

Question 39)$_{128e}$ (4 points)
Which of the following is a special ability of insertion sort?

a) It can sort most arrays in $\mathcal{O}(n)$ time.

b) It can give useful results if interrupted before it has finished.

c) It can return the elements closest to the beginning of the vector.

d) It runs in $\mathcal{O}(n)$ time if the data is already sorted.

Question 40)$_{12cc}$ (4 points)
Consider this vector: `[5 4 6 3 8 7 2]`.
The quicksort partition algorithm performs most badly if the pivot chosen is:

a) 6

b) 8

c) 4

d) 5

Question 41)$_{1291}$ (4 points)
What is the worst-case performance of quicksort?

a) $\mathcal{O}(1)$

b) $\mathcal{O}(n)$

c) $\mathcal{O}(n \lg n)$

d) $\mathcal{O}(n^2)$

Question 42)$_{1292}$ (4 points)
How can we prevent the worst-case performance of quicksort?

a) Check if the data is already sorted first.

b) Always chose a pivot in the middle of the array.

c) Chose a random pivot.

d) Always chose the first element as a pivot.

Question 43)$_{12c4}$ (4 points)

Given the following heap, show the result of deleting one element from it.

0	1	2	3	4	5	6	7
2	3	5	8	12	10		

a)

0	1	2	3	4	5	6	7
3	5	8	10	12			

b)

0	1	2	3	4	5	6	7
2	3	5	8	12			

c)

0	1	2	3	4	5	6	7
3	8	5	10	12			

d)

0	1	2	3	4	5	6	7
5	8	12	10				

Question 44)$_{12c3}$ (4 points)

Consider the following min-heap:

0	1	2	3	4	5	6	7
2	3	5	8	12	10		

What will it look like if we insert a 1 into it?

a)

0	1	2	3	4	5	6	7
2	3	5	8	12	10	1	

b)

0	1	2	3	4	5	6	7
1	3	2	8	12	10	5	

c)

0	1	2	3	4	5	6	7
2	3	5	8	12	10		1

d)

0	1	2	3	4	5	6	7
1	3	5	8	12	10		

Question 45)$_{12cd}$ (4 points)

Suppose you have the following hash table:

Pos	Content
0	
1	
2	a
3	b
4	
5	
6	

If we intert an element `c`, and the hash function tries to place it in position 2, where will the element end up if we use linear probing?

a) 1

b) 3

c) 4

d) 5

Question 46)$_{12ce}$ (4 points)

Suppose you have the following hash table:

Pos	Content
0	
1	
2	a
3	
4	
5	b
6	

If we intert an element `c`, the primary hash function tries to place it in position 2, and the secondary hash function returns 3, where will the element end up if we use double hashing?

a) 0

b) 1

c) 3

d) 5

Question 47)$_{12c1}$ (4 points)

What is the time complexity of inserting an element into a hash table that is nearly empty?

a) $\mathcal{O}(1)$

b) $\mathcal{O}(\lg n)$

c) $\mathcal{O}(n)$

d) $\mathcal{O}(n^2)$

Question 48)$_{12c2}$ (4 points)

What is the time complexity of inserting an element into a hash table that is nearly full?

a) $\mathcal{O}(1)$

b) $\mathcal{O}(\lg n)$

c) $\mathcal{O}(n)$

d) $\mathcal{O}(n^2)$

Question 49)$_{12c5}$ (4 points)

What application would be a good use for uptrees?

a) Checking if a network of computers is entirely connected

b) Building a perfect hash function

c) Sorting nodes in a graph

d) Finding the largest element of a set

Question 50)$_{12c6}$ (4 points)

Suppose we have an uptree where A points to B, B points to C, and C points to D. If we have path compression, and call `find(A)`, what will happen?

a) It will simply return A.

b) It will simply return D.

c) Nothing.

d) A,B,and C will point to D.