2

Illinois Institute of Technology
Computer Science

First Exam

CS 331 — Data Structures
Spring 2015
Friday, February 27, 2015 15:15–16:30

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

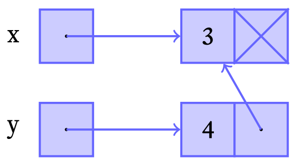
Question 3)$_{12d1}$ (4 points)

Suppose we have this Clojure code.

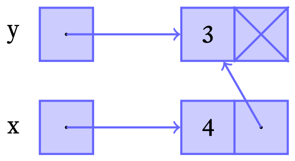
```
(def x (cons 3 nil)))
(def y (cons 4 x)
```

Which memory diagram best describes the result?

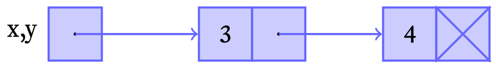
a)



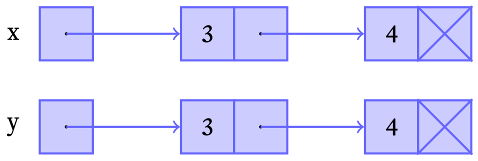
b)



c)

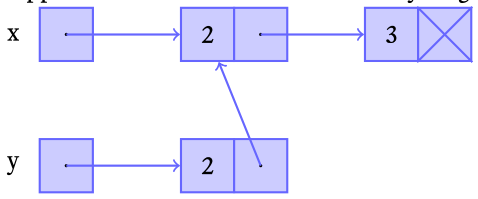


d)



Question 4)$_{12d0}$ (4 points)

Suppose we want to create this memory diagram.



Which code sequence can do this?

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

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

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

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

Question 5)$_{12cf}$ (4 points)
If you have a persistent singly linked list, where is the **least** efficient place to insert a new element?

a) the beginning

b) the end

c) it depends on whether or not you have a `last` pointer

d) all inserts are the same

Question 6)$_{1265}$ (4 points)
If you have a mutable singly linked list with a `last` pointer, which operation can be done more quickly than a mutable singly linked list without a `last` pointer?

a) insert at beginning

b) insert at end

c) delete from beginning

d) delete from end

Question 7)$_{1266}$ (4 points)
The "delete by copying" 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)$_{12d4}$ (4 points)
Suppose I need a list that will be shared among several processes, and suspect the size will change frequently. What implementation is best?

a) array list

b) persistent singly linked list

c) mutable singly linked list

d) mutable doubly linked list

Question 10)$_{12d5}$ (4 points)
If we need the functionality of a doubly-linked list in a persistent environment, what should we do?

a) Use a doubly linked list.

b) Use two singly linked lists.

c) Use a zipper.

d) Use a queue.

Question 11)$_{12d6}$ (4 points) What is the time complexity for deleting an element from a doubly linked list if you have a pointer to the containing node?

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)$_{12d8}$ (4 points)
What is the time complexity for `enqueue` 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 a two-list queue?

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

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

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

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

Question 15)$_{12d9}$ (4 points)
What is the time complexity for `pop` 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)$_{12b5}$ (4 points)
If you want to use an array/vector to implement a queue, what do you need?

a) a pointer to the first available space, and a pointer to the head of the queue

b) a counter holding the size, and a pointer to the head of the queue

c) just a pointer to the tail of the queue

d) just a pointer to the last inserted element

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)$_{12da}$ (4 points)
What is the time complexity for accessing an element of an array / a vector?

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

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

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

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

Question 20)$_{12db}$ (4 points)
What does the `transient` function do?

a) Makes a list into a zipper.

b) Makes a vector into a list.

c) Allows a vector to be updated in-place.

d) Allows a vector to be reversed.