COS 226	Algorithms and Data Structures	Fall 2011

Midterm

This test has 9 questions worth a total of 60 points. You have 80 minutes. The exam is closed book, except that you are allowed to use a one page cheatsheet. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. **Write out and sign the Honor Code pledge before turning in the test.**

"I pledge my honor that I have not violated the Honor Code during this examination."

Problem	Score
0	
1	
2	
3	
4	
Sub 1	

Problem	Score
5	
6	
7	
8	
Sub 2	

Total	

Name:

Login ID:

Precept:

P01	11	Maia Ginsburg
P01A	11	Aman Dhesi
P02	12:30	Sasha Koruga
P02A	12:30	Joey Dodds
P03	1:30	Maia Ginsburg
P03A	1:30	Joey Dodds

0. **Miscellaneous. (1 point)**

   In the space provided on the front of the exam, write your name and Princeton NetID; circle
   your precept number; and write and sign the honor code.

1. **Union find. (6 points)**

   Circle the letters corresponding to arrays that *cannot* possibly occur during the execution of
   weighted quick union.

   ```
              i:   0  1  2  3  4  5  6  7  8  9
                  ---------------------------
   A.    a[i]:    1  2  3  0  1  1  1  4  4  5

   B.    a[i]:    9  0  0  0  0  0  9  9  9  9

   C.    a[i]:    1  2  3  4  5  6  7  8  9  9

   D.    a[i]:    0  0  0  0  0  1  1  1  6  2

   E.    a[i]:    0  0  0  0  0  1  1  1  6  8

   F.    a[i]:    0  0  0  1  1  3  3  7  7  7
   ```

   (A and E are circled.)

2. **Analysis of algorithms. (6 points)**

   Suppose that you collect the following timing data for a program as a function of the input
   size N.

N	time
125	0.03 sec
1,000	1.00 sec
8,000	32.00 sec
64,000	1,024.00 sec
512,000	32,768.00 sec

   Estimate the running time of the program (in seconds) as a function of N and use tilde
   notation to simplify your answer.

   Hint: recall that $\log_b a = \lg a / \lg b$.

3. **Data structures. (9 points)**

Suppose that the Java library `java.util.LinkedList` is implemented using a doubly-linked list, maintaining a reference to the first and last node in the list, along with its size.

```
public class LinkedList<Item> {
   private Node first;             // the first node in the linked list
   private Node last;              // the last node in the linked list
   private int N;                  // number of items in the linked list

   private class Node {
      private Item item;           // the item
      private Node next, prev;     // the next and previous nodes
   }
   ...
}
```

(a) Using the 64-bit memory cost model from the textbook, how much memory (in bytes) does a `Node` object use and how much does a `LinkedList` object use to store N items? Do *not* include the memory for the items themselves but do include the memory for the references to them.

- Memory of a `Node`:

- Memory of a `LinkedList` with N items:

(b) What is the order of growth of the *worst-case running time* of each of operation below? Write down the best answer in the space provided, using one of the following possibilities.

$$1 \qquad \log N \qquad \sqrt{N} \qquad N \qquad N \log N \qquad N^2$$

addFirst(item)	*prepend the item to the beginning of the list*	
get(i)	*return the item at position i in the list*	
set(i, item)	*replace position i in the list with the item*	
removeLast()	*delete and return the item at the end of the list*	
contains(item)	*is the item in the list?*	

4. **8 sorting and shuffling algorithms. (8 points)**

The column on the left is the original input of strings to be sorted or shuffled; the column on the right are the string in sorted order; the other columns are the contents at some intermediate step during one of the 8 algorithms listed below. Match up each algorithm by writing its number under the corresponding column. Use each number exactly once.

```
navy   coal   corn   blue   blue   blue   wine   bark   mist   bark
plum   jade   mist   gray   coal   coal   teal   blue   coal   blue
coal   navy   coal   rose   gray   corn   silk   cafe   jade   cafe
jade   plum   jade   mint   jade   gray   plum   coal   blue   coal
blue   blue   blue   lime   lime   jade   sage   corn   cafe   corn
pink   gray   cafe   navy   mint   lime   pink   dusk   herb   dusk
rose   pink   herb   jade   navy   mint   rose   gray   gray   gray
gray   rose   gray   teal   pink   navy   jade   herb   leaf   herb
teal   lime   leaf   coal   plum   pink   navy   jade   dusk   jade
ruby   mint   dusk   ruby   rose   plum   ruby   leaf   mint   leaf
mint   ruby   mint   plum   ruby   rose   pine   lime   lime   lime
lime   teal   lime   pink   teal   ruby   palm   mint   bark   mint
silk   bark   bark   silk   bark   silk   coal   silk   corn   mist
corn   corn   navy   corn   corn   teal   corn   plum   navy   navy
bark   silk   silk   bark   dusk   bark   bark   navy   wine   palm
wine   wine   wine   wine   leaf   wine   gray   wine   silk   pine
dusk   dusk   ruby   dusk   silk   dusk   dusk   pink   ruby   pink
leaf   herb   teal   leaf   wine   leaf   leaf   ruby   teal   plum
herb   leaf   rose   herb   cafe   herb   herb   rose   sage   rose
sage   sage   sage   sage   herb   sage   blue   sage   rose   ruby
cafe   cafe   pink   cafe   mist   cafe   cafe   teal   pink   sage
mist   mist   plum   mist   palm   mist   mist   mist   pine   silk
pine   palm   pine   pine   pine   pine   mint   pine   palm   teal
palm   pine   palm   palm   sage   palm   lime   palm   plum   wine
----   ----   ----   ----   ----   ----   ----   ----   ----   ----
 0                                                                1
```
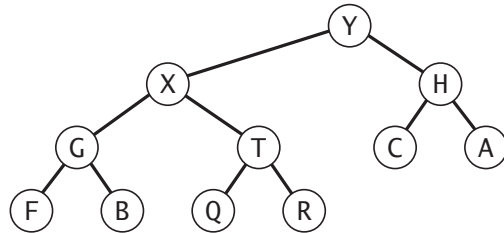
(0) Original input

(1) Sorted

(2) Selection sort

(3) Insertion sort

(4) Mergesort (*top-down*)

(5) Mergesort (*bottom-up*)

(6) Quicksort (*standard, no shuffle*)

(7) Quicksort (*3-way, no shuffle*)

(8) Heapsort

(9) Knuth shuffle

5. **Binary heaps. (6 points)**

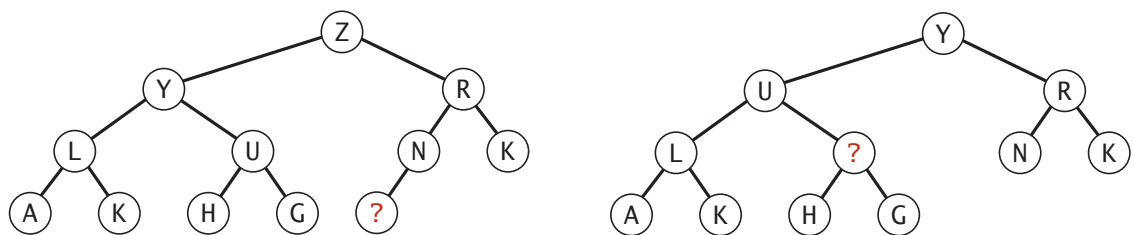   (a) Consider the following binary tree representation of a max-heap.

   Give the array representation of the heap.

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

   (b) *Insert* the key P into the binary heap above, *circling* any entries that changed.

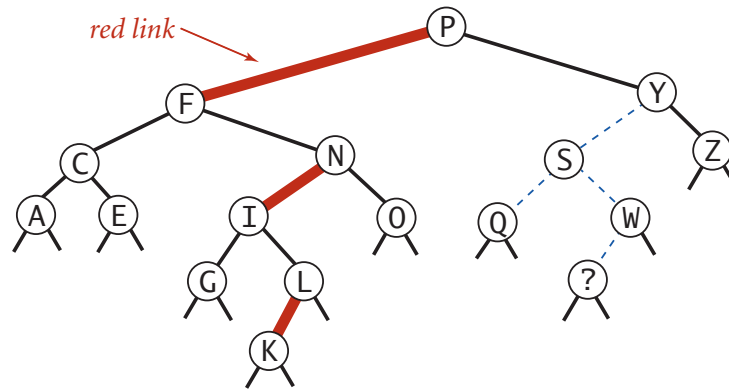| 0	1	2	3	4	5	6	7	8	9	10	11	12
–												

   (c) A *delete-the-max* operation in the binary heap at left results in the binary heap at right.

   Which of the keys below could be the one labeled with a question mark?
   Circle all possibilities.

   A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

6. **Red-black BSTs. (8 points)**

Consider the following left-leaning red-black BST. Some of the colors and key values are suppressed.

(a) Which of the keys below could be the one labeled with a question mark?
    Circle all possibilities.

    A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

(b) For each link from the left-hand column, select its possible color(s) from the right-hand column.

    ____ link between W and S   A. red

    ____ link between ? and W   B. black

    ____ link between S and Y   C. either red or black

    ____ link between Q and S

(c) How many *left rotation*, *right rotation*, and *color flip* operations would be used to insert each key below into the original red-black BST above?

	H	D	B	J
rotateLeft()	1			
rotateRight()	0			
flipColors()	0			

7. **Comparing two arrays of points. (8 points)**

   Given two arrays `a[]` and `b[]`, each containing N distinct points in the plane, design two algorithms (with the performance requirements specified below) to determine whether the two arrays contains precisely the same set of points (but possibly in a different order).

   For each algorithm, give a crisp and concise English description of your algorithm.
   Your answer will be graded on correctness, efficiency, and clarity.

   (a) Design an algorithm for the problem whose running time is linearithmic in the *worst case* and uses at most constant extra space.

   (b) Design an algorithm for the problem whose running time is linear under reasonable assumptions and uses at most linear extra space. Be sure to state any assumptions that you make.

8. **Stabbing count queries. (8 points)**

Given a collection of x-intervals and a real value x, a *stabbing count query* is the number of intervals that contain x. Design a data structure that supports interval insertions intermixed with stabbing count queries by implementing the following API:

```
public class IntervalStab
```

`IntervalStab()`	*create an empty data structure*
`void insert(double xmin, double xmax)`	*insert the interval (xmin, xmax) into the data structure*
`int count(double x)`	*number of intervals that contain x*

For example, after inserting the five intervals $(3, 10)$, $(4, 5)$, $(6, 12)$, $(8, 15)$, and $(19, 30)$ into the data structure, `count(9.1)` is 3 and `count(17.2)` is 0.

If there are N intervals in the data structure, you should support *insert* and *count* in time proportional to $\log N$ in the worst case (even if `count()` returns N).

For simplicity, assume that no two intervals contain a left or right endpoint in common and that the argument to the stabbing count query is not equal to a left or right endpoint.

Give a crisp and concise English description of your data structure.
Your answer will be graded on correctness, efficiency, and clarity.

- `IntervalStab():`

- `insert(xmin, xmax):`

- `count(x):`