

**CMSC132
Fall 2005
Midterm #2**

First Name: _____

Last Name: _____

Student ID: _____

Section time _____ **TA:** _____

Grader Use Only:

#1		(10)
#2		(10)
#3		(15)
#4		(15)
#5		(15)
#6		(15)
#7		(10)
#8		(10)
Total		(100)
Honors		(10)

I pledge on my honor that I have not given or received any unauthorized assistance on this examination.

Your signature: _____

General Rules (Read):

- This exam is closed book and closed notes.
- If you have a question, please raise your hand.
- Total point value is 100 points.
- Answer True/False questions by circling the T or F at the end of the question.
- Answer multiple-choice questions by circling the letter (e.g., a, b) at the front of each choice.
- Answer essay questions concisely using 1 or 2 sentences. Longer answers are not necessary and are discouraged.
- **WRITE NEATLY.** If we cannot understand your answer, we will not grade it (i.e., 0 credit).

Problem 1 Networking (20 pts)

A. (4 pts) Network architecture

a. What is a protocol?

A formal description of formats and rules

b. Why are protocols useful for a network?

They describe messages formats and actions taken needed by computers talking to each other on a network

c. Why is the OSI network model designed as 7 separate layers?

To reduce the complexity of the network

d. List 3 examples of programs used at different layers of the internet.

DNS server, Telnet, FTP, TCP, , IP, device drivers for network card

e. What is the difference between an IP address and a URL?

IP address specifies a computer on the network, URL specifies a resource (e.g., web page) in addition to the computer

B. (4 pts) Reliability

a. What is the difference between reliable and unreliable network connections?

A reliable network connection will provide notification of error if data does not arrive, and ensure data arrives in order if it does arrive

b. Which of the following layers of the network model may be unreliable?

i. Physical

ii. Data-link

iii. Transport

iv. Network

c. Why do we not simply make all parts of the network reliable?

Reliability requires greater overhead

C. (4 pts) Packets vs. Connections

a. Packets are more reliable than connections T or F

b. Packets are more expensive than connections T or F

c. Packets are simpler than connections T or F

d. Which protocol may be viewed as a connection? UDP or TCP

D. (4 pts) Client-server model

a. Servers can only serve one client T or F

b. A client contacts a server first T or F

c. How do clients distinguish between multiple servers at the same IP address?

Port number

d. List examples each 3 server and 3 client programs.

Servers – Apache, POP server, IMAP server... Clients – IE, Firefox, Outlook...

E. (4 pts) Java support for networking

a. What are sockets?

Application layer view of network connection

b. What is the main difference between the Socket and ServerSocket classes?

Socket class is for client, ServerSocket class is for server

c. What is the main difference between the Socket and DatagramSocket classes?

Socket class supports TCP, DatagramSocket supports UDP protocol

d. Why is there no DatagramServerSocket class?

UDP protocol does not distinguish between clients & servers

e. In Java, why does data from a network appear similar to data from a file?

Because the Socket class methods view network data as IOStreams

f. What are applets?

Java programs downloaded from network that run with restrictions in the JVM

Problem 2 Files, Scanner, and Generics in Java

F. (4 pts) Files & Scanner

a. Files may be viewed as streams when opened using either the FileReader and FileInputStream classes. What is the difference between the two types of streams?

View data as a stream of text or as a stream of bytes

b. What is the motivation for using the Scanner class to process text input?

Standard interface for recognizing more complex forms of data such as ints, floats, doubles, etc. (rather than as strings)

c. Write code to show how the scanner may be used to read in a series of alternating numbers (ints) and names (Strings) from a text file (starting with a number).

```
class Scanner {  
    Scanner ( ... ) { ... }  
    boolean hasNext() { ... }  
    String next() { ... }  
    int nextInt() { ... }  
    String nextLine() { ... }  
}
```

```
myFileReader() {  
    try {  
        BufferedReader f = new BufferedReader( new FileReader( filename ));  
  
        // insert code to read in numbers and names from file  
  
    } catch (IOException e) {  
        System.out.println(e.getMessage());  
    }  
}
```

G. (4 pts) Generics

- a. What is the motivation for using generic types?
Java programs downloaded from network that run with restrictions
- b. Classes in the Java Collection Framework support generic types in Java 1.5. Show how to modify the following code to declare a LinkedList class for Strings using generics, then add & get a String object from the list.

```
// previously
LinkedList myList = new LinkedList( );
myList.add( new String("test") );
String s = (String) myList.get(0);
```

Problem 3 Algorithmic Complexity (20 pts)

H. (4 pts) Algorithmic complexity

- a. What is algorithmic complexity?
Amount of resources required by algorithm with respect to problem size
- b. List 2 reasons benchmarking is better than analyzing complexity
Measures performance for a particular combination of hardware, software, and input data. Yields precise performance results (e.g., 10 minutes)
- c. What is the difference between best case, worst case, and average case?
Minimum, maximum, and typical number of steps required by algorithm
- d. What is a recurrence relation?
Value of a function is defined in terms of the function at other points

I. (4 pts) Big-O notation and complexity

- a. What does the Big-O notation represent?
Upper bound on the number of steps required by algorithm
- b. Why are $O(2n+4)$ and $O(n)$ considered equivalent?
The number of steps required by each algorithm grows at the same rate with respect to the problem size
- c. Why are $O(n^2)$ and $O(n)$ not considered equivalent?
The number of steps required by each algorithm grows at a different rate with respect to the problem size
- d. What are some simple rules for calculating Big-O notation?
Ignore constant terms, constant coefficients, lower-order (less complex) terms
- e. Sort the following complexity categories from least to most complex
 - i. Constant $O(1)$
 - ii. Cubic $O(n^3)$
 - iii. Exponential $O(2^n)$
 - iv. Linear $O(n)$
 - v. Logarithmic $O(\log(n))$
 - vi. Quadratic $O(n^2)$

Constant, logarithmic, linear, quadratic, cubic, exponential

f. How are the following complexity categories similar?

$O(n^2)$, $O(n^4)$, $O(n^3)$, $O(n^{12})$, $O(n^{99})$

These algorithms all take polynomial time -- $O(n^k)$ for a constant value of k

J. (4 pts) Calculating Big-O functions

What is the asymptotic complexity of the function $f()$ below (using big-O notation) when the complexity of $f()$, and $g()$ are as shown?

$f(n)$ {
 $g(n)$;
 $h(n)$;
}

- a. $h(n) = n+1$, $g(n) = 2n$ $f(n) = O(\mathbf{n})$
b. $h(n) = 3n$, $g(n) = 4n^2$ $f(n) = O(\mathbf{n^2})$
c. $h(n) = \log(n)$, $g(n) = 5\sqrt{n}$ $f(n) = O(\mathbf{\sqrt{n}})$
d. $h(n) = 8\sqrt{n}$, $g(n) = 2n$ $f(n) = O(\mathbf{n})$

K. (4 pts) Finding critical regions

Calculate the asymptotic complexity of the code snippets below (using big-O notation) with respect to the problem size n:

- a. `for (i = 1; i < n; i=i+2) {` $f(n) = O(\mathbf{n})$
 ...
}
- b. `for (i = 1; i < n; i=i*2) {` $f(n) = O(\mathbf{\log(n)})$
 ...
}
- c. `for (i = 0; i < n; i++) {` $f(n) = O(\mathbf{n^2})$
 `for (j = 1; j < n; j=j+2) {`
 ...
 }
}
- d. `for (i = 0; i < n-2; i++) {` $f(n) = O(\mathbf{n^2})$
 `for (j = 0; j < 100; j=j+2) {`
 `for (k = 1; k < 3*n; k++) {`
 ...
 }
 }
}

- ```

 }
e. for (i = 0; i < n; i=i*2) { f(n) = O(n log(n))
 for (j = 1; j < n; j++) {
 ...
 }
}

f. for (i = 0; i < n-2; i++) { f(n) = O(n2)
 for (j = 0; j < n; j=j*2) {
 for (k = 1; k < 5000; k=k*5) {
 ...
 }
 }
 for (j = 0; j < n; j=j+1) {
 ...
 }
}

```

#### **Problem 4 Recursive Algorithms (20 pts)**

##### L. (4 pts) Recursion

- Describe the difference between an iterative and recursive algorithm?  
**An iterative algorithm reapply actions in a loop, a recursive algorithm reapply actions by calling itself**
- What are the 2 main parts of a recursive algorithm?  
**Base case, recursive step**
- Name 4 requirements for recursive algorithms to work correctly.  
**Base case recognized & solved correctly, recursive step solves 1 or more simpler subproblems, & combines solutions to yield correct overall solution**
- Name 2 advantages & 2 disadvantages of recursive algorithms  
**Advantages – simpler algorithm, handles recursive data structure,  
 Disadvantages – more function calls, runs faster, uses less memory**

##### M. (4 pts) Legality of recursive code

For each of the following codes, describe what result is returned when foo(n) is invoked. If no result is returned, explain why.

- ```
int foo (int n) {
    return foo(n-1);
}
```

foo(n) = **no result, missing base case**
- ```
int foo (int n) {
 if (n == 1) return 1;
```

foo(n) = **1 (for n > 0)**

- ```

        return foo(n-1);
    }

```
- c. `int foo (int n) {` `foo(n) = n (for n > 0)`
 `if (n == 1) return 1;`
 `return 1+foo(n-1);`
`}`
- d. `int foo (int n) {` `foo(n) = no result, recursion does not`
 `if (n == 1) return 1;` `solve simpler problem`
 `return foo(n);`
`}`
- e. `int foo (int n) {` `foo(n) = 2n-1 (for n > 0)`
 `if (n == 1) return 1;`
 `return foo(n-1)+f(n-1);`
`}`

N. (4 pts) Writing recursive code (you may use helper functions)

- Write a recursive function to search an unsorted array for a number k
`public static boolean findNum(int k, int[] array)`
- Write a recursive function to calculate the sum of an array of ints
`public static int sumArray(int[] array)`
- Write a recursive function to determine whether an array of ints is sorted (ascending)
`public static boolean sortedArray(int[] array)`

Problem 5 Linear Data Structures (20 pts)

O. (4 pts) Taxonomy & properties

- Describe the main difference between linear and hierarchical data structures
1-to-1 vs. 1-to-many relationships between elements
- Describe the main difference between a linked list and an array
Nodes in linked list reference successor nodes, nodes in array do not reference other nodes but are located by array index
- Describe the main difference between a queue and a stack
First-in first-out vs. First-in, last-out
- Describe a circular linked list
Linked list where last node refers to head of list (instead of null reference)

P. (4 pts) Given the following Java class definition for a singly linked list

```

Class Node {
    int myValue;
    Node next;
}

```

```

}

Class LinkList {
    Node head;          // first node in list
    Node find(int k) { ... }
    void insertHead(Node n) { ... }
    void insertTail(Node n) { ... }
    void removeTail() { ... }
}

```

Write the following code:

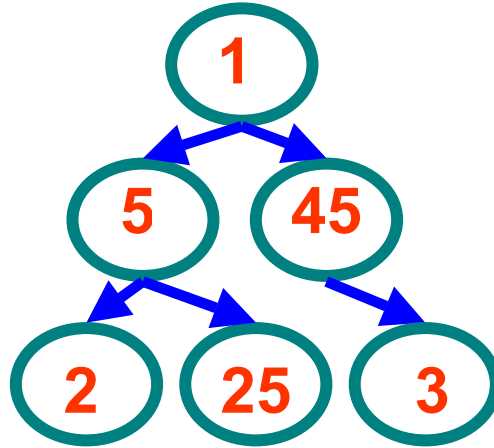
- find(k) – Find a node with myValue = k in a linked list
- insertHead(n) – Insert node n at the head of a linked list
- insertTail(n) – Insert a node n at the tail of a linked list
- removeTail() – Delete the node at the tail of a linked list
- Which 2 methods in the LinkList class can be used to implement a queue?
insertHead(), removeTail()
- Which 2 methods in the LinkList class can be used to implement a stack?
insertTail(), removeTail()

Problem 6 Trees (20 pts)

- Q. (4 pts) Binary search trees (BST)
- What is the key property of a binary search tree?
Values in left subtree < value at root, values in right subtree > value at root
 - On average, what is the complexity of doing an insertion in a binary search tree?
O(log(n))
 - On average, what is the complexity of doing a find in a binary search tree?
O(log(n))
 - What is the worst-case complexity of doing a find in a binary search tree?
O(n)
 - What can cause worst-case behavior in a binary search tree?
Unbalanced / degenerate binary trees
- R. (4 pts) Binary search trees examples
- Draw the binary search tree created when the following values are inserted in order:
6, 5, 2, 8, 10, 3
 - Given the previous BST, draw the tree created when the node 5 is removed
- S. (4 pts) Traversals
- What is a tree traversal?
Visiting all the nodes in a tree
 - What is the difference between a depth-first and breadth-first traversal?

Depth-first traversal visits entire subtree first, breadth-first traversal visits nodes in order of their distance from the root

- c. Pre-order, in-order, and post-order are all depth-first traversals T or F
- d. Pre-order traversals are faster than post-order traversals T or F
- e. For the following binary tree, provide an example of a
 - i. Preorder traversal **10, 5, 2, 25, 45, 30** or **10, 45, 30, 5, 25, 2** or ...
 - ii. Postorder traversal **2, 25, 5, 30, 45, 10** or **25, 2, 5, 30, 45, 10** or ...
 - iii. Breadth-first traversal **10, 5, 45, 2, 25, 30** or **10, 45, 5, 30, 2, 25** or ...



T. (4 pts) Given the following Java class definition for a binary tree

```

Class Node {
    int myValue;
    Node left;
    Node right;
}

Class Tree {
    Node root; // root node in tree
    Node find(int k) { ... }
    int treeHeight() { ... }
    void preorder() { ... }
}
  
```

Write the following code:

- a. find(k) – Use a recursive algorithm to find a node with myValue = k in tree
- b. treeHeight() – Use a recursive algorithm to find the height of a tree
- c. preorderPrint() – Use a recursive algorithm to print myValue for each node using a preorder traversal of the tree, starting from the root

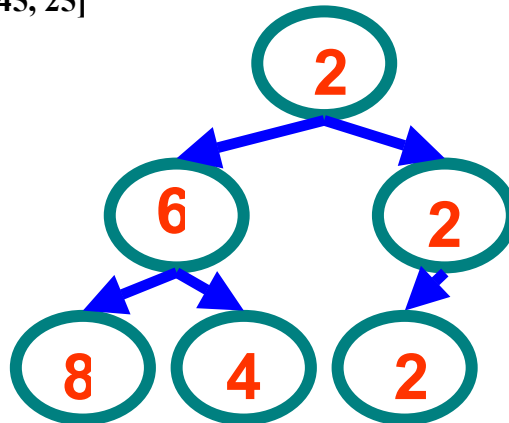
Problem 7 Heaps (20 pts)

U. (4 pts) Properties & characteristics

- a. What are two key properties of a heap?
Complete binary tree where value at node is smaller or equal (can also be larger or equal) to values in subtrees
- b. What operation(s) supported by binary search trees are not supported by heaps?
find largest key, find key with value k
- c. On average, what is the complexity of doing an insertion in a heap?
 $O(\log(n))$
- d. On average, what is the complexity of doing a *getSmallest()* in a heap?
 $O(\log(n))$
- e. What is the worst-case complexity of doing a *getSmallest()* in a heap?
 $O(\log(n))$
- f. How can heaps be stored in a compact array representation?
Assign node locations in array based on formula for array index
- g. Why are heaps used to implement priority queues?
Can efficiently select items with higher priority first

V. (4 pts) Examples

- a. Draw the heap created when the following values are inserted in order:
6, 5, 2, 8, 10, 3
- b. Given the previous heap, draw the heap produced when the smallest value is removed
- c. Show the tree representation of the following heap (in array representation)
A, B, C, D, E, F
- d. Show the array representation of the following heap (in tree representation)
[2, 6, 22, 8, 45, 25]



Problem 8 Maps & Hashing (20 pts)

W. (4 pts) Sets & maps

- a. What is the difference between a set and a map?
A map is a set that has one or more keys associated with objects in the collection, where keys may be used to retrieve objects
- b. What is the difference between a map and a hash table?

A hash table is a map where keys can be transformed into a number and used as an index into the hash table

c. How are maps useful?

Maps provide an interface for retrieving data based on keys

d. Given the following TreeMap API, show how to write code to construct a TreeMap storing String objects for actor names with movie titles. The TreeMap should allow movie names to be used to look up lead actor names (assuming each movie has a single leading actor)

```
public class TreeMap {
    TreeMap() { ... }
    Object get( Object key ) { ... }
    Object put( Object key, Object value ) { ... }
    Object remove( Object key ) { ... }
}
```

```
TreeMap myDB = new TreeMap();
```

```
void addMovie( String leadActor, String movie ) {
    ... // write code to add leading actor name for movie to TreeMap
}
```

```
String findLeadActor( String movie ) {
    ... // write code to find String for leading actor name for movie
}
```

X. (4 pts) Hashing

a. What is a hash function?

A function to generate a number from a key

b. What is a desirable property of hash functions?

Numbers generated from keys are spread out uniformly in a range

c. What is a perfect hash function?

Every key is assigned a unique number

d. What is a collision?

Identical number generated for different keys

e. What is the difference between open addressing and chaining (bucket hashing)?

The hash table directly store elements (open addressing), instead of storing references to lists of elements (bucket hashing)

f. What happens when an open addressing hash table is close to full?

Number of steps required to perform operations no long require O(1) time

g. What is the relationship between the hashCode() and equals() methods in Java?

Two objects with different hashCode() must not be equal(). Two objects that are equal() must have the same hashCode().