CMPS 212 Intermediate Programming with Data Structures

FINAL EXAM
Winter 2002-2003

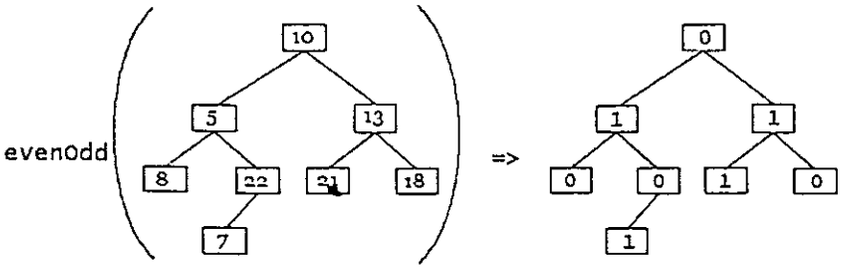
Department of Mathematics & Computer Science
American University of Beirut

- **Each question is worth 20 points for a total of 100**
- **Please read the questions carefully BEFORE you start.**
- **All questions must be asked in the first hour of the exam.**

(1) Consider the definition of the Binary Tree abstract data type shown below:

```
class BinTree {
        Integer     data;     // data field
        BinTree     left;     // link to left subtree
        BinTree     right;    // link to right subtree
        public void doubleAll() { ... }
        public BinTree evenOdd() { ... }
        ...
}
```


Complete the methods **doubleAll**() and **evenOdd**(). The method **doubleAll**() doubles all the data elements in the tree. The method **evenOdd**() returns a binary tree that where every node has either a 0, if the node at that position in the original binary tree was even, and 1 if the node at that position was odd (you can assume you have two boolean functions **isEven**(Integer i) and **isOdd**(Integer i).



(2) (a) Write a pure (non-destructive) recursive Java method '**public int sum(Vector v)**' that takes a Vector v, and returns the sum of all the Integer elements in v.

(b) Using the method defined in (a), write a pure (non-destructive) recursive Java method '**public Vector sumAll(Vector v)**' that takes a Vector of vectors, v, and returns a vector containing the sum of every vector in v.

(3) Write the pseudo code of heap sort (of a heap stored in an array), and show why heap sort has $O(n \log n)$ complexity.

(4) (i) The following Java statements will not compile properly. Fix the (syntactic!) problem(s) in the code


```
public Vector test(Vector v) {
      Vector r = new Vector();
      Vector copy = v.clone();
      if ( copy.size() == 0 ) return v;
      elseif (copy.size() > 2 ) {
            for ( i = 0; i< copy.size(); i++ ) {
                  Integer n = v.elementAt(i);
                  if ( n > 100 ) return v;
                  r.addElement( new Integer(2*n) );
            }
      }
}
```

(ii) Show the complete trace of the following function call 'funny 3 4', where funny is defined as follows:

```
funny n 0 = 0
funny n m = n + funny n (m-1)
```

What does the function 'funny' do?

(iii) What is the complexity of the following function?

```
simple  0 = 1
simple  n = sum[1..n] + simple(n/3) + simple(n/3) + simple(n/3)
```

Explain how you arrived at your result.

(5) i) Answer the following questions with **True** or **False**

a. The quadratic probing method to handling collisions in a Hashtabledoes not guarantee to find a slot in the table, even if the table is not full. [_____]
b. The time complexity of MergeSort can, in the best case, be $O(n)$. [_____]
c. Every user defined type is an abstract data type (ADT) [_____]
d. Since Vectors behave like dynamic objects (i.e., we can add and remove elements as needed), data in vectors must be stored in linked lists [_____]
e. A properly designed ADT has all data members declared private [_____]

ii) Fill in the blanks

a. _____ refers to the fact that all data members in an object must be hidden, while _____ refers to the fact that the implementation details of the all the public methods must be hidden.
b. In performing depth-first search we need to store elements that are yet to be explored in a _____, while in performing breadth-first search such elements are stored in a _____.
c. The _____ of an object is a snapshot of the values of all the data members at a specific point in time.
d. To perform a binary search for x on a sorted array, we can first compare x with _____. Subsequently, we can search in _____ and ignore _____.
e. In the average case, the total number of calls QuickSort( ) makes on an input vector of size n is _____ while the work inside _____ yielding a total of _____.

AMERICAN UNIVERSITY
LIBRARY
OF BEIRUT

CMPS 212 – Final Exam - Fall 2004

Name: _____ Username:_____ Section___

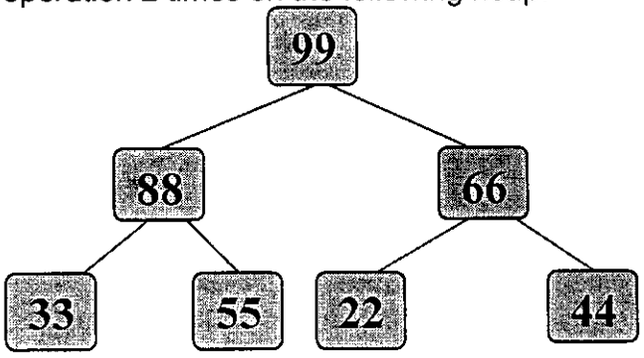
1. a) (5 pts) Take the following numbers and insert them into a heap in the given order. Show the content of the heap array after you perform this operation:

33 63 22 54 91 83 25

Answer:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	... ...

b) (5 pts) Show the content of the heap array after you perform the remove operation 2 times on the following heap.



Answer:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	... ...

2. (5 pts) Consider the following 2 recursive methods on an integer array data with n integers:

```
public static int mystery1(int[] data, int n) {
    int sum;
    if (n <= 0)  return 0;
    else {
        sum = mystery1(data, n-1);
        return sum;
    }
}

public static int mystery2(int[] data, int n) {
    int sum;
    if (n <= 0)  return 0;
    else {
        if (data[n-1] % 2 == 0)
            sum = mystery2(data, n-1) + 1;
        else
            sum = mystery2(data, n-1);
        return sum;
    }
}
```

What is the final return value of each of these methods if they are called with the following parameters : data = {2, 2, 3, 3, 3, 4, 4, 4, 4}, n = 9?

Answer for mystery1( ):

Answer for mystery2( ):

3. (5 pts) A hash table is created to store integer keys using a hash function h(k). The current state of the hash table is shown below. All keys were inserted without any collisions.

INDEX	0	1	2	3	4	5	6	7	8	9	10	11	12
KEY		14		29		83	45				62	11	38
HAS_BEEN_USED	F	T	F	T	F	T	T	F	F	F	T	T	T

a) What is h(k)?
Answer:

b) At what position will the key 23 be stored in the hash table using h(k) above if linear probing is used to resolve collisions?
Answer:

c) At what position will the key 23 be stored in the original hash table (which does not include 23) if double hashing is used to resolve collisions, assuming $h_1(k) = h(k)$ and $h_2(k) = 1 + (k \bmod 11)$?
Answer:

4. (5 pts) What is the time complexity (Big-Oh notation) of the following algorithm, which takes as input an unsorted integer array *arr [ ]* of size *n*. Assume heapSort(), insertionSort() and mergeSort() are functions for Heap Sort, Insertion Sort and Merge Sort respectively. Also assume that n is an even integer. Explain your answer.

```
mySort(int[] arr) {
    sort the lower half of arr using heapSort();
    sort the upper half of arr using insertionSort();
    merge both halves by using the merge operation as defined for mergeSort();
}
```

5. (5 pts) Given A[ ] [ ] which is an n-by-n matrix of integers. What the complexity of the given code as a function of n?

```
int i = 0;
int j = 0;

while (i< n && j < n)
{
        if (0 == (A[i][j] % 2))
                i++;
        else
                j++;
}
```

Answer:

6. (4 pts) Given a singly linked list with references to its head and tail. Which one of the following set of statements moves the head node to the end of the list.

a) tail.link = head;
b) tail.link = head; tail = head; head = head.link; tail.link = null;
c) tail.link = head; head.link = null; head = head.link; tail = head;
d) tail.link = head; head = head.link; tail.link = null;

7. (5 pts) A binary search tree T has the following pre-order sequence: 42137658. Draw the tree T.

8. (3 pts) In terms of computational complexity, what data structure would be most suitable for the following scenario? You are required to store *n* ClassA objects where ClassA has a *key* attribute to use for comparison. The objects will be presented to you in ascending order and must be stored in ascending order. After all of the objects have been stored, the data structure may have to be searched numerous times for particular ClassA objects. No additional objects will need to be added to the data structure. Pick one of the following answers then justify your answer.

(a) an array of length 1000 of ClassA references
(b) an array of length *n* of ClassA references
(c) a linked list of ClassA references
(d) a stack of ClassA references
(e) a queue of ClassA references

Justification:

9. (3 pts) How can a computer system ensure that a user has entered the correct password without actually storing the password itself anywhere?

10. (3 pts) What does this method compute?

```
public int foo(int n)
{
        if (n <= 1)
                return 1;
        else
                return 5 + foo(n-1);
}
```

a) foo(n) = the nth Fibonacci number
b) foo(n) = 5*n
c) foo(n) = n^5
d) foo(n) = 5^n
e) none of the above

11. (4 pts) Provide an implementation of the recursive method rec() defined as:
rec(0) = 37
rec(1) = 73
rec(n) = n / (rec(n-1) + rec(n-2))

12. (2 pts) Name one application of priority queues:

_____

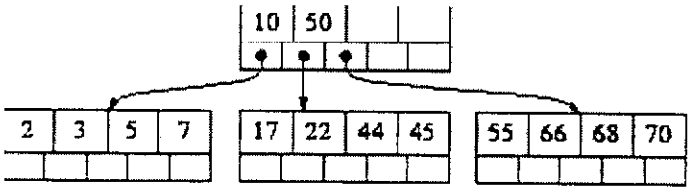
13. (6 pts) Draw the Control Flow Graph of the following code where the label denotes the basic block number:

```
b1: int x = 1;
b1: int y = 2;
b1: int z = 3;

b1: if (1 == x)
    {
b2:     x = 2;
b2:     if (2 == y)
        {
b3:         y = 3;
b3:         if (3 == z)
            {
b4:             z = 4;
            }
        }
    }

b5: System.out.println("z = " + z);
```

14. (5 pts) Show the state of the following B-Tree after inserting 6 in it.



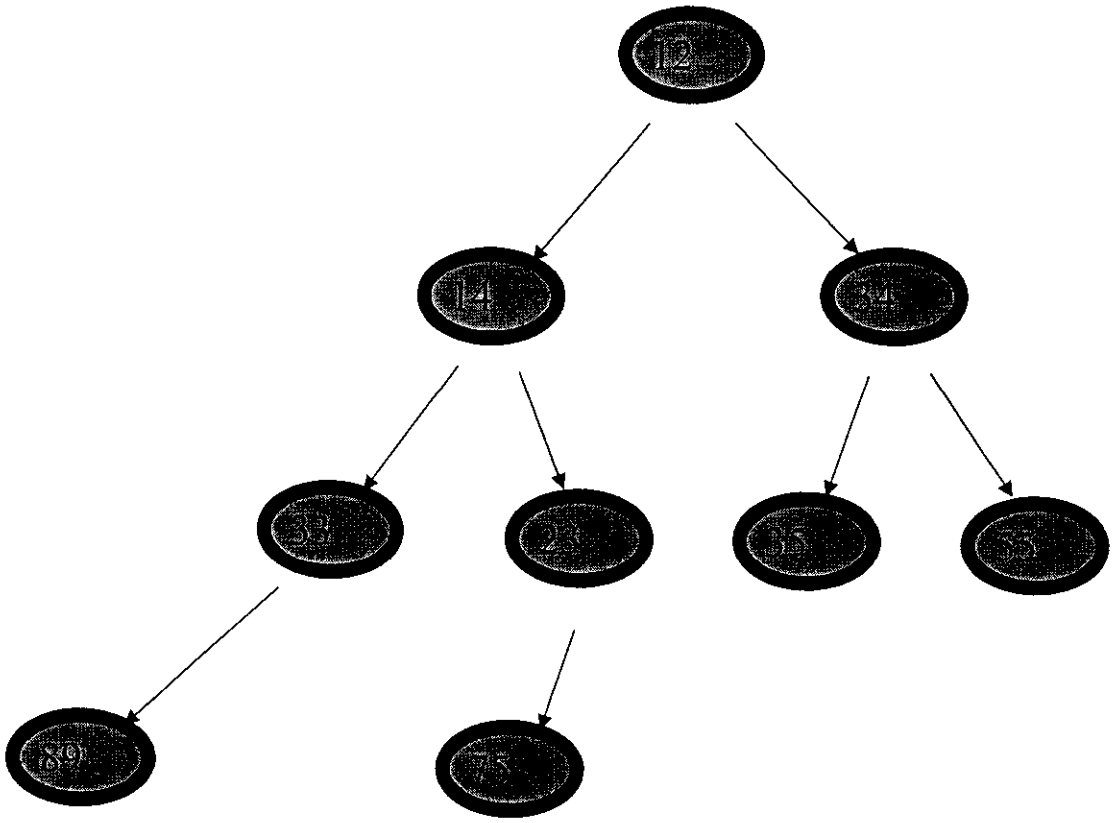
15. (4 pts) Starting with an empty binary search tree, show its state after inserting the following nodes in the given order: 56 34 42 38 1 2

16. (5 pts) Given a binary search tree built by inserting the following nodes in the given order: 99 75 40 39 8 3 2.

a) What is the height of the tree?

b) Why is it not desirable to have a tree that looks as such?

17. (5 pts) Show the order of traversal of this tree, given post-order traversal



Answer:

18. (6 pts) Which one of the following 2 statements is true or false:
a) Any problem that can be solved with recursion can be solved iteratively, but the opposite is not always true. True ___ False ___

b) Any problem that can be solved iteratively can be solved with recursion, but the opposite is not always true. True ___ False ___

c) Computing the factorial of a number and the Tower of Hanoi problem are both easily solved iteratively. True ___ False ___

19. (5 pts) What is the output of the following snippets of Java code?

```
boolean b = false;
if (b = false) {
        System.out.println("I am in");
}
else {
        System.out.println("I am out");
}
```

Output:
I am __

```
boolean b = false;
if (b = true) {
        System.out.println("I am in");
}
else {
        System.out.println("I am out");
}
```

Output:
I am __

20. (10 pts) A hash table is defined using an array of 100 IntNode references, so that collisions can be handled by using chained hashing, as follows:

```
public class Table {
    private int manyItems;
    private IntNode[] keys;

    public Table() {
        keys = new IntNode[100];
        manyItems = 0;
    }

    private int hash(int key) {
        return key % 100;
    }

    // other Table methods
}
```

Write Java code for the following Table methods below. You may assume that IntNode is a class that defines a singly-linked integer node with the methods getData, setData, getLink, setLink, and a default constructor.

(a) public void put(int key)
//Inserts the key into the appropriate "chain" of the hash table.
// You may assume that the key is not a duplicate.

(b) public boolean containsKey(int key){
// Returns true if the key is in the hash table, or false otherwise.