CQI Report
Fundamentals Committee
Fall 2008

Prepared by:

Olac Fuentes (committee chair)
Marine Ceberio
Vladik Kreinovich
Mary Kay Roy
Eric Freudenthal

CS 2401 Review – Fall 2008

Prepared by Olac Fuentes

Course Description:

This is the second course for students majoring in Computer Science. Students will learn about fundamental computing algorithms, including searching and sorting; elementary abstract data types including linked lists, stacks, queues and trees; and elementary algorithm analysis.

Knowledge and abilities required before the students enter the course:

Students are assumed to be comfortable programming in Java. Students should be able to code basic arithmetic expressions, define simple classes, use strings, code loops and conditional statements, write methods, create objects from classes, invoke methods on an object, perform basic text file input and output, and use arrays.

Learning Outcomes

Level 3 Outcomes: Synthesis and Evaluation

Identify, implement and use:

a. Multi-dimensional arrays.
b. Lists implemented as arrays or linked lists.
c. Stacks.
d. Queues.
e. Binary trees and binary search trees.
f. Simple hashes

Level 2 Outcomes: Application and Analysis

a. Use Big-O notation to express the best-, average- and worst-case behaviors of an algorithm
b. Explain the structure and use of activation records
c. Determine the best, average and worst-case behaviors of an algorithm
d. Assess time and space trade-offs in algorithms.
e. Explain, code, and use quadratic and O(n log n) sorting algorithms
f. Implement recursive algorithms over natural numbers, lists, and trees
g. Define and use classes, subclasses and inheritance.
h. Implement a simple graphical user interface
i. Perform string manipulation and simple parsing
j. Implement and use multidimensional arrays
k. Describe the importance of encapsulation and information hiding
l. Implement applications and simulations of the data structures identified above.
m. Implement simple sequential and binary search algorithms
n. Implement quadratic sorting algorithms
o. Describe memory allocation of integers, real numbers, arrays and objects

Level 1 Outcomes: Knowledge and Comprehension

a. Explain basic and introductory-level notions of a virtual machine
b. Explain the concept of polymorphism
c. Use class browsers and related tools
d. Identify class hierarchies
e. Recognize the standard terms associated with particular data structures e.g. head/tail, push/pop/peek

Instruments

Assessment was done using 3 partial exams, a final exam, and 10 laboratory assignments. The table shows the mapping from outcomes to assessment instruments and the average grade, normalized to the 0-1 range, obtained by students in that instrument.

Assessment Mapping Table

Outcome	Level	Test 1	Test 2	Test 3	Other	Final Exam	Result
Explain basic and introductory-level notions of a virtual machine	1						Explained, assigned as homework, not tested
Explain the concept of polymorphism	1	Q10:6 A: 0.99			Lab 1 A:0.75		Met
Use class browsers and related- tools	1				Lab Assignments 2-10 TA Assesment:80%		Met
Identify class hierarchies	1				Lab Assignments 2-10 TA Assesment:80%		Met
Recognize the basic terms associated with particular data structures e.g. head/tail, push/pop/peek	1		Q8:8 A: 0.71 Q9:8 A: 0.47			Q7:10 A: 0.80 Q8:10 A: 0.51 Q9:10 A: 0.55	Met
Use Big-O notation to express the best-, average- and worst-case behaviors of an algorithm	2			Q1:12 A:0.72 Q2:9 A:0.69		Q1:12 A: 0.89 Q13:10 A: 0.71 Q14:10 A: 0.66	Met
Explain the structure and use of activation records	2						Explained, not tested
Determine the best, average and worst-case behaviors of an algorithm	2			Q1:12 A:0.72 Q2:9 A:0.69		Q14:10 A: 0.66	Met
Assess time and space trade-offs in algorithms.	2				Lab7 A:0.75 Lab 9 A:0.90		Met
Explain, code, and use quadratic and O(n log n) sorting algorithms	2			Q3:8 A:0.99 Q4:8 A:1.00	Lab7 A:0.75 Lab 9 A:0.90		Met
Implement recursive algorithms over natural numbers, lists, and trees	2	Q5:5 A: 0.96 Q6: 8 A: 0.28 Q7:10 A: 0.28			Lab 2 A:0.81 Lab 3 A:0.85 Lab 4 A:0.88 Lab 5 A:0.80 Lab 6 A:0.90	Q3:8 A: 0.76 Q4:9 A: 0.84	Met
Define and use classes,	2				Lab Assignments		Met

subclasses and inheritance.					2-10 TA Assesment:75%		
Implement a simple graphical user interface	2						No covered
Perform string manipulation and simple parsing	2	Q8:12 A: 0.32			Lab 5 A:0.80 Lab 7 A:0.75	Q5:5 A: 0.95	Met
Describe the importance of encapsulation and information hiding	2				Lab Assignments 1-10 TA Assesment:80%		Met
Implement applications and simulations of the data structures identified above.	2				Labs 1 to 10 A:0.83		Met
Implement simple sequential and binary search algorithms	2						Covered, not tested
Implement quadratic sorting algorithms	2				Lab 6 A:0.90 Lab 9 A:0.90		Met
Describe memory allocation of integers, real numbers, arrays and objects	2	Q12: 8 A: 0.73					Met
Multi-dimensional arrays.	3	Q2:10 A: 0.80 Q3:10 A: 0.82 Q4: 10 A:0.76 Q9:10 A:0.58 Q11:8 A:0.37			Lab 1 A:0.75 Lab 3 A:0.85	Q1:12 A: 0.89 Q2:10 A: 0.64	Met
Lists implemented as arrays or linked lists.	3		Q1:10 A: 0.79 Q2:10 A: 0.76 Q3:12 A: 0.55 Q4:12 A: 0.54 Q5:12 A: 0.56 Q6:10 A: 0.45 Q7:12 A: 0.38		Lab 6 A:0.90 Lab 7 A:0.75	Q6:10 A: 0.70 Q7:10 A: 0.80	Met
Stacks	3		Q8:8 A:0.75 Q10:20 A:0.36		Lab 8 A:0.70	Q8:10 A: 0.51	Met
Queues	3		Q9:8 A: 0.47 Q10:12 A: 0.31		Lab 8 A:0.70	Q9:10 A: 0.55	Met

Binary trees and binary search trees.	3			Q5: 8 A: 0.92 Q6: 9 A: 0.87 Q7: 9 A: 0.83 Q8: 12 A: 0.73 Q9: 12 A: 0.58 Q10:12 A: 0.20 Q11: 12 A: 0.49	Lab 10 A:0.95	Q10:4 A: 0.79 Q11:10 A: 0.68 Q12:10 A: 0.57	Met
Simple hashes	3			Q:12 8 A: 0.96			Met

Performance analysis by instructor

Table 1 presents the assessment results for each course outcome. "Met" in the last column indicates that for a question the mean is 70% or better. Overall, the course was successful in attaining the main educational outcomes. In particular, most level 3 outcomes were tested multiple times to ensure compliance and were met.

We make following observations and recommendations:

1. Outcome level 1, *Explain basic and introductory-level notions of a virtual machine,* was explained in class and was the subject of a homework assignment, but was not tested. This was deemed sufficient since it's a level one outcome and the topic will be explained in detail in advanced courses.
2. Outcome level 2, *Explain the structure and use of activation records* was not tested. The material was explained in class when recursion was introduced, but not tested. We suggest this outcome be moved to CS3 Data Structures and presented when recursion is covered in more depth.
3. Outcome level 2, *Implement simple sequential and binary search algorithms* was not tested. The material was explained and used as an example when big-O notation was introduced, but it was not tested, as the code is provided in the textbook. The instructor believed that there was little to be gained by having the students just re-type that code. We recommend the outcome be modified to "*Describe and analyze sequential and binary search algorithms".*
4. Outcome level 2, *Implement a simple graphical user interface* was described in the lab by the T.A., but not tested. We recommend user graphical interfaces be required as parts of at least two of the labs in the course.

Comments by Committee:

Strengths of course:

The majority of learning outcomes was met.
Labs: varied topics at the right level of depth
Exams: cover outcomes at appropriate level of depth. Interesting types of questions: Given a complexity in big-Oh and parameters, implement an algorithm of the given complexity.

Issues:

Some instructors prefer labs with specific topics, while labs in this course are for the most part lists of shorter tasks to be implemented. The committee should discuss which approach works best for training the students (the answer to this question should also involve data about their motivation level, which can affect the retention of students).

Opportunities for improvement:

Some outcomes not met, as already mentioned in analysis by instructor.

Recommendations for change to course delivery:

Lower failing grades are desirable.

Recommendations for changes to program:

Move Outcome level 2, *Explain the structure and use of activation records* to CS3, as suggested by instructor.

Replace Outcome level 2 *Implement simple sequential and binary search algorithms* by *Describe and analyze sequential and binary search algorithms,* as suggested by instructor, since the code is presented in most textbooks and is also readily available online.

Assessment Instruments

CS2401
Fall 2008
Exam 1

1. (8 points) The following method was written with poor programming style and documentation Rewrite it to fix these problems.

```
public static int k(int kgb){
                        // This method is recursive
                    if (n > 0)  // comparing n and 0
return 0; //
        return n*k(n-1);
                        }
```

2. (10 points) Write a method that receives a 1D array of integers A and an integer n and counts and returns the number of times n appears in A.

3. (10 points) Consider the following method:

```
public static int [] cs(int [] A){
        int [] C = new int[A.length];
        C[0] = A[0];
        for(int i=1; i<A.length;i++)
                C[i] = C[i-1]+ A[i];
        return C;
}
```

Trace the execution of cs(A), where A = {2,4,5,7,8}.

4. (10 points) Write a method that receives a 2D array of integers A and an integer n and returns the row where n first appears in A, or -1 if n does not appear in n. For example, if A= {{1,2}, {4,5}, {5,8}}, and n = 5, your method should return 1, and if A= {{1,2}, {4,5}, {5,8}}, and n = 9, your method should return -1.

5. (5 points) When is a recursive solution to a problem preferable to an iterative solution?

6. (8 points) Consider the following method:

```
public static void s(String S, int n) {
   if (n>0){
        s(S+"0",n-1);
        s(S+"1",n-1);
    }
    else
        System.out.println(S);
 }
```

Trace the execution of s("",3)

7. (10 points) Write a recursive method that computes the sum 1/n + 1/(n-1) + 1/(n-2) + ... + ½ + 1/1.

8. (12 points) Write a method that receives a String S and returns the lower case vowel that appears in S with the highest frequency. For example, if S = "El Paso, Texas", your program should return 'a'. You may break ties arbitrarily.

9. (10 points) Write a recursive method that prints the contents of a 1D array in reverse.

10. (6 points) Explain the following concepts:
    a. Inheritance
    b. Polymorphism
    c. Method overriding

11. (8 points) What is the output of the following code fragment?

```
int [][] x = {{1,2,3},{4,5,6},{7,8,9}};
for (int i=0;i<x.length;i++)
   x[i] = x[0];
printArray(x);
x[2][0]= -1;
printArray(x);
```

12. (8 points) What is the output of the following code fragment?

```
void p(int [] k){
   int [] m = k;
   m[1] = -10;
}
.
.
.
int [] x ={1,2,3};
p(x);
printArray(x);
```

CS2401
Fall 2008
Exam 2

1. (10 points) What will be the output of the following code fragment:

```
Node x = null;
for (i=0;i<50;i=i+10)
        Node x = new Node(i,x);
printList(x);
Node y= x.link;
x = y.link;
printList(x);
printList(y);
Node z = new Node(100,y);
printList(z);
y =null;
printList(z);
while (z.link!=null)
        z=z.link;
printList(z);
```

2. (10 points) Consider the method c written below. Show a box trace of the execution of c(x,0), where x points to a reference-based list that contains a sequence of nodes with values [2,-4,8,10,-15,-8]

```
void c(cNode x, int k){
        if (x!=null){
                if (x.info<k)
                        x.info =k;
                c(x.link,k);
        }
}
```

3. (12 points) Write the method that takes as parameters *h*, a reference to the head of a linked list of integers, and *n*, an integer, and prints the first *n* elements of the list, or the whole list if it has less than *n* elements. Use only node operations to access the list.

4. (12 points) Write a method that receives a reference to the head of a list of characters and returns a String containing the same characters as the list in the same order.

5. (12 points) Write an iterative and a recursive method to compute and return the length of a linked list.

6. (10 points) Write a method to print the contents of a circular linked list.

7. (12 points) Write the method that concatenates two doubly-linked lists.

8. (8 points) Consider a stack implemented as a reference-based list. Show the configuration of the list after each push or pop operation.

```
Stack s = new Stack();
for (int i=0;i<5;i++){
        s.push(i);
        s.push(i+10);
}
for (int i=0;i<5;i++)
        j = s.pop();
```

9. (10 points) Consider an implementation of a queue that uses an array of size 5. Show the array and the values of front, back, and count after each enqueue or dequeue operation.

```
Queue q = new Queue();
for (int i=1;i<6;i++)
        q.enqueue(i);
j = q.dequeue();
k = q.dequeue();
m = q.dequeue();
q.enqueue(j);
q.enqueue(k);
q.enqueue(m);
```

10. Write two versions of the method inQueue(n) that determines if element n is in the queue Your methods should be written as follows:
    a. (10 points) As an additional method inside the queueList class that implements queues with circular reference-based lists (recall that they keep a reference to the last element of the queue).
    b. (10 points) As a method outside the class that defines queues, that is, accessing the queue only through the ADT queue operations.

11. Write two versions of the method inStack(n) that determines if element n is in the stack Your methods should be written as follows:

    a. (10 points) As an additional method inside the stackArray class that implements stacks with arrays.
    b. (10 points) As a method outside the class that defines stacks, that is, accessing the stack only through the ADT stack operations.

CS2401
Fall 2008
Exam 3

1. (12 points) In terms of n, what are the big "O" running times of the following code fragments? Explain your answers.

c.
```
void m(int [] A, int n){
    System.out.println(A[n]);
}
```

d.
```
void m(int [] A, int n){
    int temp =0;
    while (n>=temp){
        for(int i=0;i<temp;i++)
            System.out.println(A[i]);
        temp++
    }
}
```

e.
```
void m(int [] A, int n{
    if(n>0){
        System.out.println(A[n]);
        m(A,n/2);
    }
}
```

f.
```
void m(int [] A, int n{
    if(n>0){
        System.out.println(A[n]);
        m(A,n-1);
    }
}
```

2. (9 points) Write methods that receive an array *A* and an integer *n* and take the specified big-O running times.

g. $O(n^3)$
h. $O(2^n)$
i. $O(n \log n)$

3. (8 points) Trace the execution of bubble sort with the following input: 15,30,10,8, 35,31,22,40.

3. (8 points) Trace the execution of mergersort with the following input: 15,30,10,8, 35,31,22,40.

4. (8 points) Show the configuration of an initially empty binary Search Tree after inserting the sequence of elements : 15,30,10,8, 35,31,22,40.

5. (9 points) Write the outputs produced by a) preorder, b) inorder, and c) postorder traversals using the tree from question 5 as input.

6. (9 points) Show the tree from the question 5 after deleting the sequence of elements 30, 31, and 15. Draw the tree after each deletion.

7. (12 points) Write a method that receives a binary tree *T* and prints the contents of all the nodes in *T* that have exactly on child.

8. (12 points) Write a method that receives a binary search tree *T* and an integer *k* and prints the level where *k* is found in *T* or -1 if *k* is not in *T* (recall that the root has level 0, the children of the root have level 1, and so on).

9. (12 points) Write a method that receives a binary tree *T* and prints the value of the smallest element in *T*. You may assume that *T* is not empty.

10. (12 points) Write a method that receives a binary **search** tree *T* and prints the value of the smallest element in *T*. You may assume that *T* is not empty. Make your method as efficient as possible.

11. (8 points) Consider a hash table that resolves collisions by chaining and uses the hash function h(x) = x % 5. What is the configuration of the table after inserting elements with keys: 15, 30, 10, 8, 35, 31, 22, and 40, in that order?

CS2401
Fall 2008
Final Exam

1. (10 points) Write a method that receives a 1D array of integers A and an integer k and determines if A contains at least one element that is greater that k. (2 points) If A contains n elements, what is the big-O running time of your method, in terms of n?

2. (10 points) Write a method that receives a 2D array of integers A and returns a 1D array containing the sums of the rows in A. For example, if $A = \{\{10, 20, 30\}, \{50, 100, 200\}\}$, your method should return a 1D array containing $\{60, 350\}$.

3. (8 points) Write a recursive method that receives two characters a and b and a positive integer k and prints a k times, followed by b k times. For example, if a ='x', b = 'y', and k =3, your method should print xxxyyy.

4. (9 points) Consider the method written below.

```
public static void p1(int [] A, int k){
    if (k<A.length){
        A[k] = A[k-1]+ A[k];
        p1(A,k+1);
    }
}
```

   (a) Let A be initially $\{1, 2, 3, 5, 7, 9, 11\}$. Trace the execution of p1(A,1).

   (b) In general, what does p1(A,1) do?

   (c) Let n be the size of A (that is, A.length), what is the "big-O" running time of p1, in terms of n?

5. (10 points) Write a method that receives a String S and returns S in reverse.

6. (10 points) What will be the output of the following code fragment:

```
iNode x = null;
iNode y;
iNode z;
for(int i=0;i<10;i=i+2)
    x = new iNode(i,x);
printList(x);
y = x.link;
z = y.link;
printList(y);
printList(z);
y.link = z.link;
printList(x);
y = null;
printList(x);
```

7. (10 points)Write a method that receives a reference to the head of a linked list of integers L and an integer k and returns the position where k is in L (starting with 0), or -1 if k is not in L. (2 points) If L contains n nodes, what is the big-O running time of your method, in terms of n?

8. (10 points) Write a method that receives an item k and a queue Q as inputs and returns the position where k is in Q, from front to back, or -1 if k is not in Q (thus your method should return 0 if k is at the front of the queue). You must only use the elementary operations of the ADT queue to access Q and must not alter the contents of Q.

9. (10 points) Write a method that receives a stack S as input and deletes the largest element in S, leaving the rest unchanged. You may assume S is not empty. Also, you may only access S using the operations associated with the ADT Stack.

10. (4 points) Draw a binary search tree that contains the keys 1, 2, ..., 14, 15 and has the maximum possible height. Draw a binary search tree that contains the same keys and has the minimum possible height.

11. (10 points) Write a method that receives a reference to the root of a binary tree T and prints the contents of all the internal nodes in T. Recall that an internal node is a node that has at least one child, that is, an internal node is any node that is not a leaf.

12. (10 points) Write a method that receives a reference to the root of a binary search tree T and two integers k and m and prints all the elements in T that are greater than k and less that m (assume $k < m$). Make your method as efficient as possible (that is, make no unnecessary recursive calls).

13. (10 points) Write methods that receive an array A and an integer n and take the specified big-O running times.

    (a) $O(log\, n)$
    (b) $O(n)$
    (c) $O(n\, log\, n)$
    (d) $O(n^2)$
    (e) $O(2^n)$

14. (10 points) For each big-O function in the previous question, give an example of an algorithm covered in this class that has the specified big-O running time.

CS2401 Fall 2008
Lab 1

Due September 5, 2008, 5:00 p.m.

Your first lab consists of writing the following methods that work on 1D arrays:

1. Two methods to print the contents of arrays. The first method should work for arrays of type double and the second should work for integers. Make sure your methods have the same name (this is an instance of method overloading, a very useful feature of Java.)

2. Two methods to allow users to input arrays from the keyboard. Again, use overloading to make the first method work with doubles and the second with integers.

3. Two methods to allow users to fill-up arrays with uniformly distributed random numbers. Your methods should receive the minimum and maximum values and fill the arrays with randomly-chosen values that are greater or equal to the minimum and less or equal to the maximum. Use overloading as before.

4. A method that receives an integer and prints, in English, the sequence of digits in that integer. Your method must use arrays of strings to store the names of the digits. A sample run of your method would be as follows:

   Enter an integer
   15644
   The integer 15664 contains the following digits:
   one five six four four

5. A method that receives an array of integers in the 0 to 9 range and determines if the array contains each of the single digits (0 to 9) at least once. For example if your method receives the array A={0, 3, 4, 5, 0, 1, 2, 3} it should return false and if it receives the array A={1, 4, 6, 2, 3, 4, 5, 7, 9, 8, 0, 5, 1} it should return true. (Hint: use an array of Booleans)

6. A main method that illustrates the correct behavior of the methods in numbers 1 to 5.

7. (Extra Credit) Repeat problem 4, but now, instead of printing to the screen the sequence of digits, have the computer read them aloud to the user. Consult Java sound resources on the web for information about how to do this - it's cool and it's not too hard!

   Here's a possible sample output of your main method:

```
Testing Method 1 a
{1.1, 2.2, 3.3, 4.4, 5.5}

Testing Method 1 b
{1, 2, 3, 4, 5}

Testing Method 2 a
Enter size of array of doubles to be input:
>> 4
Enter 4 numbers
>> 4 5 6 7
{4.0, 5.0, 6.0, 7.0}

Testing Method 2 b
Enter size of array of integers to be input:
>> 8
Enter 8 integers
>> 45 89 7 10 2 36 6444 5
{45, 89, 7, 10, 2, 36, 6444, 5}

Testing Method 3 a
Enter size of array of doubles to be randomly generated:
>> 4
Enter minimum and maximum values:
>> 0 100
{9.559843287214331, 21.041310595035988, 84.60780683107396, 62.17104822610812}

Testing Method 3 b
Enter size of array of integers to be randomly generated:
>> 10
Enter minimum and maximum values:
>>0 100
{39, 96, 60, 68, 17, 47, 62, 15, 35, 12}

Testing Method 4
Enter integer to be broken into digits:
>> 478966
The integer 478966 contains the following digits:
four seven eight nine six six

Testing Method 5
Enter size of array of integers to be tested for the presence of every digit:
>> 20
Enter 0 to enter array manually, 1 to generate randomly:
>> 1
The array
{9, 5, 0, 6, 8, 8, 0, 5, 4, 8, 7, 9, 1, 0, 7, 3, 6, 4, 4, 7}
Does not contain every digit from zero to nine
```

CS2401 Fall 2008
Lab 2

Due September 12, 2008, 5:00 p.m.

There are a number of problems, known collectively as "random walk" problems, which have been of longstanding interest to the mathematical community. Some of these problems, although extremely difficult to solve analytically, can be solved by 2401 students (hopefully!) using simulation. One of them is as follows:

A cockroach is placed on a given square in the middle of a tile floor in a rectangular floor of size *n* by *m* tiles surrounded by walls. The cockroach wanders randomly from tile to tile throughout the room. Assuming that he may move with equal probability to any of the neighboring four tiles (North, South, East, West) with equal probability, how many moves will it take him to touch every tile on the floor at least once? If the cockroach bumps into a wall, its position is not changed, but the move should be counted.

Your lab consists of implementing a simulation of this problem. Your program must prompt the user to enter the number of rows and columns in the tile floor and the initial position of the cockroach; it will then simulate the cockroach's random walk and terminate when every tile has been visited at least once. Your program must then output the initial parameters, the number of moves taken, and a 2-D array showing the number of times each tile was visited. Also, write methods to make sure that the sum of the elements in your array is equal to the number of moves the cockroach made and that the minimum element in your array is 1 (why?).

A sample output of your program would look follows:

```
Enter number of rows: 5
Enter number of columns: 8
Enter starting row: 4
Enter starting column: 2
It took the cockroach 278 moves to visit every tile
The number of times it visited each tile is as follows:
    8   9   6   6  10  13  14  15
    6   4   6   6   7  13  16   8
    4   2   3   4   2   5  13   7
    4   4   1   1   4   8  14  18
    1   1   1   1   2   6  12  13
The sum of the elements in the array is: 278
The minimum of the elements in the array is: 1
```

CS2401 Fall 2008
Lab 3

Due September 19, 2008, 5:00 p.m.

1. Write a recursive method called *print_nums* that receives as parameter an integer *n*. The method will print to the screen 1 in the first line, 1 2 in the second line, and so on until it prints all the numbers from 1 to *n* in a line. Your method should call another recursive method to print each line. For example. If *n* =4, the method should print to the screen:

   1
   1 2
   1 2 3
   1 2 3 4

2. Write a recursive method to reverse the contents of a 1-dimensional array of integers.

3. Write a recursive method to determine if a 1-dimensional array of integers is sorted in ascending order.

4. Solve programming exercise 1 from page 925 in the textbook.

5. Solve programming exercise 4 from page 926 in the textbook.

CS2401 Fall 2008
Lab 4

Due September 26, 2008, 5:00 p.m.

1. The subset sum problem is an important problem in computer science. Given a set of positive integers S and a target integer n, the goal is to find if there is a subset of S whose sum is n. For example, if S = {1, 3, 6, 7} and n = 13, the answer is true, because the sum of {6, 7} adds up to 13, but if n =5, the answer is false because there is no subset of S that adds up to 5. Write a program that prompts the user to enter a set of positive integers S and an integer n and uses a recursive method to determine if there is a subset of S whose sum is n. Notice that you don't need to show the set of integers adding up to n, you just need to determine if it exists.

2. Do exercise 15, from pages 928-930 in the textbook. To solve this problem, we strongly suggest that you look at the example on pages 916 to 921.

CS2401 Fall 2008
Lab 5

Due October 7, 2008, 5:00 p.m.

1. Implement the kth-smallest algorithm discussed in class and test is using randomly generated arrays of integers. Also, for every recursive call, print the contents of the array being processed and the value of k.

2. Write a method that receives a string S and returns true if S is a palindrome.

3. Some characters of the Spanish language do not exist in English; these are the accented vowels (á,é,í,ó,ú), and "eñe" (ñ). These characters can create problems for some English-only systems. To avoid these problems, one can replace the accented vowels by their un-accented counterparts and ñ by n. Write a method that performs this replacement. For example, if your method receives the String "Juárez", it should return "Juarez", and if it receives "Cañón", it should return "Canon".

CS2401 Fall 2008
Lab 6
Due October 15, 2008

Your lab assignment consists of implementing two algorithms (described below) to sort lists of integers implemented as a reference-based lists. Your program must prompt the user to select the length of the list, whether the elements will be entered manually or generated randomly, and the choice of algorithm to use. As the elements are entered they must be stored in a reference-based list (you may consider storing them in the inverse order in which they are entered). After that, the selected method must be called to sort the list, and finally you should display the resulting sorted list and the time it took to sort the list.

Algorithm A starts with the original list of elements and an initially empty sorted list, and repeatedly finds the node that contains the element with the maximum value in the original list and moves it to the sorted list, until the original list is empty

```
SortListA(L)
        SortedList = empty list
        while L is not empty
                x = node with maximum element in L
                remove x from L
                add x at the beginning of SortedList;
        return SortedList
```

Algorithm B, takes the first element of the list as pivot, and splits the list into three lists; the first list contains the elements that are smaller than the pivot, the second contains the pivot itself, and the third contains the elements that are greater or equal to the pivot. Then the algorithm recursively sorts the first and the third lists, and finally it concatenates the first list, now sorted, the list that contains the pivot, and the third list, also sorted.

```
SortListB(L)
        If length(L) >1
                let f be the first node in L
                remove f from L
                split L into three lists as follows:
                  L1 that contains all the elements that smaller that the element in f
                  L2 that contains (only) f
                  L3 that contains the elements of L that are greater or equal to the element in f
                L1 = SortListB(L1);
                L3 = SortListB(L3)
                L = concatenate(L1,L2,L3);
        return L
```

Extra Credit: Modify Algorithm B to implement the kth-smallest algorithm described in class.

CS 2401 Fall 2008
Lab 7
Reference-based Lists and Strings
Due Friday, October 24

Instructions

Your task is to implement the class LString, which uses reference-based lists of characters to implement standard operations on strings. An LString will be implemented using a singly-linked list, where each node contains a character.

Each node on the list is defined by the following class:

```
public class cNode{
	public char info;
	public cNode link;

	public cNode(char c){
		info = c;
		link = null;
	}

	public cNode(char c, cNode x){
		info = c;
		link = x;
	}
}
```

An LString, which consists of a list of cNodes, is defined as follows:

```
public class LString{
	private cNode head;

	public LString(String S){
		head = buildList(S);
	}

	private cNode buildList(String S){ // Builds a list of nodes containing the characters of S
		if (S.length()==0)
			return null;
		else {
			cNode x = new cNode(S.charAt(0),buildList(S.substring(1,S.length())));
			return x;
		}
	}
}
```

You should extend the LString class to include the following operations:

1. void setCharAt(int i,char c) // Sets character at position i in the string to ch . It generates an
   // exception if i is greater than the length of the string minus one

2. void insertCharAt(int i,char c) // Insert character c at position i. It generates an
   // exception if i is greater than the length of the string
3. void deleteCharAt (int i)// Deletes character at position i. It generates an
   // exception if i is greater than the length of the string minus one
4. void print() // Prints the string
5. LString copy() // Builds a copy of the string
6. int length() // See page 620
7. char charAt(int i) // See page 620
8. int indexOf(char ch) // See page 620
9. int compareTo(LString L) // See page 620
10. int compareTo(String S) // See page 620
11. void toUpperCase() // Replaces each lowercase letter by the equivalent uppercase letter
12. void toLowerCase()// Replaces each uppercase letter by the equivalent lowercase letter

To test your work, implement a main method that illustrates the behavior of each of your methods. For example, the code fragment:

```
LString S = new LString("UTEP Miners");
S.print();
System.out.println(S.length());
System.out.println(S.charAt(5));
System.out.println(S.indexOf("n"));
System.out.println(S.compareTo(S));
```

should output exactly the same results as the following code fragment:

```
String U = "UTEP Miners";
System.out.print(U);
System.out.println(U.length());
System.out.println(U.charAt(5));
System.out.println(U.indexOf("n"));
System.out.println(U.compareTo(U));
```

CS2401 Fall 2008
Lab 8

Due November 5, 2008, 5:00 p.m.

1. The subset sum problem is an important problem in computer science. Given a set of positive integers S and a target integer n, the goal is to find if there is a subset of S whose sum is n. For example, if S = {1, 3, 6, 7} and n = 13, the answer is true, because the sum of {6, 7}, which is a subset of S, is 13, but if n =5, the answer is false because there is no subset of S that adds up to 5. Write a program that prompts the user to enter a set of positive integers S and an integer n and uses a stack to determine if there is a subset of S whose sum is n and prints the subset is it exists.

2. Exercise 19, from page 1233 in the textbook.

CS2401
Fall 2008
Lab 9

Due Friday, November 14, 2008.

Your lab assignment consists of implementing the following algorithms to sort reference-based lists of integers:

1. Selection sort
2. Bubble sort
3. Mergesort
4. Quicksort

Your program must prompt the user to select the length of the list, whether the elements will be entered manually or generated randomly, the choice of algorithm he/she wants to use, and whether the sorted list must be printed. As the elements are entered, they must be stored in a reference-based list (you may consider storing them in the inverse order in which they are entered). After that, the selected method must be called to sort the list, and finally, you must display the time it took to sort the list, the number of comparisons performed by the algorithm, and, if requested, the resulting sorted list,.

CS2401 Fall 2008
Lab 10

Due December 4, 2008

1. Implement a method to build a binary search tree (BST) of integers. Your method must allow building a tree with randomly generated elements or with elements entered by the user from the keyboard. Make sure that all the elements in your tree are unique; if a number that is already in the tree is entered, it should be ignored.

2. Implement a recursive method to print the nodes in your tree in ascending order.

3. Implement a recursive method to print the nodes in your tree in descending order.

4. Implement a method to print the nodes in your tree ordered by level, starting with the root. Hint: use a queue.

5. Implement a method that receives an integer n and displays the contents of the parent and the children of the node that contains n, if n is in the tree.

6. Implement methods to:
    a. Print the smallest element in the BST
    b. Print the largest element in the BST
    c. Compute the sum of all the nodes in the BST
    d. Count the number of nodes in the BST
    e. Count the number of leaves in the BST
    f. Count the number of nodes in the BST that have only one child.
    g. Count the number of nodes in the BST that have two children.

    (If implemented correctly, the sum of the results of e), f), and g) should be equal to the result of d).

CS2402: Data Structures and Algorithms

Course Description:

The definition and implementation of abstract data types; representation of data using sets, lists, trees, and graphs; the design and implementation of traversal, search, and sort algorithms; and the space and time analysis of algorithms.

Learning Outcomes

Note: my letter grades are as follows:

- A is 85% and above;
- B is 75% and above;
- C is 65% and above;
- D is 55% and above;
- F is below 55%.

Level 3: Synthesis and Evaluation

A. Specify data structures and operations associated with abstract data types

- In each lab assignment
- MT3 Ex. 2 (70%)
- Final Part II (67%), Part III.8 (75%)

Comment: Met.

B. Define the signature and pre- and post-conditions for operations of an abstract data type

- In each lab assignment, especially lab 4 (79%).
- Mostly in Quiz 2 Q1 (64%),

But also in all questions expecting algorithms:

- Quiz 3 Q3 (67%), Q4 (75%), Quiz 4 Q1 (75%), Q2 (50%), Quiz 5 Q2 (56%)
- MT1 Ex. 5 (40%)
- MT2 Ex. 1 (65%)
- MT3 Ex. 2 (70%), Ex. 3 (75%)
- Final Ex. 5 (80%), Ex. 9 (62%)

Comment: Met. The grades for MT1 were low but lower grades are usually observed at the first MT. Similarly, students' performance in quizzes is in general lower (absences, lack of timely work on class topics).

C. Given a scenario, describe the abstract data types that could be created

- In each lab assignment
- MT3 Ex. 2 (70%)
- Final Part II.5 (80%), Part III.8 (75%)

Comment: Met.

D. Implement binary and binary search trees, using pre-, post-, or in-order traversals as appropriate for a given situation

- Lab 4 (79%)
- Quiz 4 (70%)
- Quiz 5 Q1 (88%)
- MT1 Ex. 5 (40%)
- MT3 Ex. 2 (70%)
- Final Part II.5 (80%)

Comment: Met. Same comment as for B for MT1.

E. Judge which data model (list, tree, graph, or set) is appropriate for solving a problem

- Labs 2 (75%), 3(77%), 4 (79%), and 5 (76%)
- MT2 Ex. 1 (30%)
- MT3 Ex. 2 (70%)
- Final Part III.8 (75%)

Comment: Met. Although students' performance was well below reasonable at MT2, they significantly improved by the end of the semester (see averages for MT3 and Final).

F. Justify the choice of a data structure to solve a problem based on issues such as time, and space, of the data structure

- Mainly in Labs 2 (75%), 3 (77%), and 5 (76%)

Comment: Met.

G. Judge which implementations are best suited for an application that requires a list data model: lists, circular lists, circular queue, or generalized list

- Lab 3 (77%)

Comment: Met.

H. Judge whether an array or linked implementation is best suited for an application that requires a data model

- Labs 3 (77%) and 5 (76%)
- MT3 Ex. 2 (70%)

Comment: Met.

I. Judge which graph representations (adjacency list, adjacency matrix, edge list) are appropriate for solving a problem

- Lab 5 (76%)
- Final Part IV (65%)

Comment: Met.

J. Develop algorithms that are based on depth- and breadth-first traversals of general trees, binary trees, and graphs

- Labs 2 (75%) and 4 (79%)

- Quiz 3 Q4 (75%)
- Quiz 4 (70%)
- MT1 Ex. 5 (40%)
- MT2 Ex. 1 (65%)
- Final Part III.8 (75%), Part III.9.4 (50%)

Comment: Met. Same comment as for B for MT1. The lower performance at the Final is due to the lack of time to complete the exam. Many students did not answer Part III.9.4.

K. Judge which sort algorithm (insertion, selection, mergesort, heapsort, quicksort, radix) is appropriate for solving a problem

- Lab 1 (78%)
- Many non-graded in-class exercises

Comment: Met.

L. Judge which search algorithm and data structure is appropriate for solving a problem

- Lab 4 (79%)
- MT2 Ex. 4 (86%)

Comment: Met.

M. Implement a recursive solution to a problem

- Labs 1 (78%) and 2 (75%)
- MT2 Ex. 1.2 (40%)
- Final Part IV.9.4 (50%)

Comment: Partially met. Students still have hard time using recursion when it has to be their own "creation". They did much better in labs than in exams though, which indicates that, given more time, they tend to overcome their problems.

Level 2: Application and Analysis

A. Categorize algorithms based on programming strategy, i.e., divide-and-conquer, greedy, backtracking, and dynamic programming strategies

- Labs 2 (75%) and 5 (76%)

Comment: Met.

B. Analyze iterative and recursive algorithms with respect to time and space

- Lab 1 (78%)
- Quiz 2 Q3 (90%)
- Quiz 5 Q1 (88%)
- MT1 (60%)
- MT2 Ex. 1.2 (50%)
- MT3 Ex. 3.4 (75%)
- Final Part I (65%), Part III.7 (72%)

Comment: Met.

C. Describe the applications for a dictionary/map ADT, e.g., the application of a symbol table

- Covered in class with exercises, not in any graded material

D. Give representations for and operations on a binary tree, general tree, threaded tree, heap, binary search tree, B-tree, quadtree, and graphs

- Lab 4 (79%)
- Quiz 2 Q3 (90%)
- Quiz 3 Q4 (75%)
- Quiz 4 (70%)
- Quiz 5 Q2 (65%), Q4 (75%)
- MT1 Ex. 5.1 (85%)
- MT2 Ex. 4 (86%)
- MT3 (75%)
- Final Part III (70%)

Comment: Met.

E. Determine the order for a B-tree based on memory issues

- Covered in non-graded homework.

F. Apply graph algorithms for determining shortest paths (Dijkstra' s and Floyd's algorithms), minimal spanning tree (Prim's and Kruskal's algorithms), transitive closure (Floyd's algorithm), and topological sort

- Lab 5 (76%)
- Final Part IV (65%)

Comment: Met. But there is room for improvement.

G. Select an appropriate sorting algorithm for a given situation and defend the selection

- Lab 1 (78%)
- Not covered in exams but extensively studied in class

Comment: Met.

H. Explain differences and similarities among approaches for resolving collisions in hash tables, e.g., linear probing, quadratic probing, double hashing, rehashing, chaining

- MT2 Ex. 3 (79%)

Comment: Met.

I. Apply design methods and other problem-solving strategies. Examples might include (but are not limited to) functional decomposition, design patterns, top-down design, abstraction, CRC

- Lab 2 (75%)
- MT2 Ex. 1 (40%)

Comment: Partially met. This was not very much emphasized in class, which can explain the lower performance of students on related questions.

Level 1: Knowledge and Comprehension

Level 1's outcomes are those in which the student has been exposed to the terms and concepts at a basic level and can supply basic definitions. The material has been presented only at a superficial level. On successful completion of this course, students will be able to:

A. Describe the characteristics of static, stack, and heap allocation
- Covered in class, not covered in any graded material.

B. Explain issues related to disk read/write time
- Covered in class, not covered in any graded material

C. Define strategies for balancing a binary search tree
- Quiz2 ex. 3 (90%)
- Quiz 5 ex. 1 (88%), ex. 2 (50%), ex. 4 (65%)

Comment: Met.

D. Define the algorithms for implementing B-tree operations
- Assigned as a non-graded homework.

E. Define the procedure for conducting an external sort
- Covered in class, partially covered in lab (Lab1 on sorting algorithms – 78%).

Comment: Met.

Recommended laboratory assignments:

The following list represents a set of suggested assignments for this course. It is not intended to be comprehensive and may be modified at the instructor's discretion.

1. **Implement a discrete simulation to model queuing systems**
   - Lab 3: discrete event simulation of a drive-through.
   - Result: 77% average

2. **Write a parser that stores the results in a data structure, allowing the user to query the structure**
   - Lab 3 requires analyzing data stored from past simulations.
   - Lab 4 on genealogical trees required students to parse a "family" file and build corresponding trees that were to be queried afterwards.
   - Results: 77% (lab3); 79% (lab4).

3. **Write a program in which the central data structure is a graph**

- Lab 5 required students to use graphs to model computer networks.
- Result: 76%.

4. **Write a program that collects empirical data (e.g., number of collisions in a hash table or execution time for a sort algorithm) and analyze the results from the program**
   - Lab 1 on sorting algorithms included a theoretical as well as an experimental analysis on sorting algorithms' complexity.
   - Lab 2 on Hitori required students to experimentally analyze the time complexity of solving algorithms.
   - Lab 3 on simulation required students to collect data in order to determine the best setting/schedule of tellers at the drive-through.
   - Results: 78% (lab1), 75% (lab2), 77% (lab3).

Comments by Instructor:

All required material was covered, although not necessarily included in graded assignments (e.g., B-trees, static, stack, and heap allocation).

Overall, by looking at the grades, and from discussions with the TA and Peer-Leaders, it appears that students have difficulties in:

- Topics that are abstract and /or theoretical; e.g., complexity (time, space, ...)
- Implementation: although not particularly reflected on the grades, but reported from TA, and also shown on all exams containing algorithms to trace, students still have troubles designing and understanding simple pseudo-code for instance. During Fall 08 we put a special emphasis on writing pseudo-code and developing algorithms (included in many exams and quizzes). We also asked students to turn in pseudo-code of their labs prior to their final submission of their code in order to force them to practice.

As reported from labs,

- Students still struggle with programming in java.
- Although we insisted on it even more than usual over the whole semester (since it was a conclusion of the previous assessment report), students still struggle with testing: they do not understand why it is so crucial and have troubles designing sound testing strategies. They do not value this stage of their labs, which is affecting their grades.
- They also lack writing skills. This semester, we made sure to always provide a detailed outline of the expected reports (for each lab). Nevertheless, most of them do not value or understand the importance of being able to clearly communicate their work and results, and turned in very poorly written and incomplete reports.

Actions to take:

1. Include B-trees in graded assignments, and more generally grade systematically all items of the outcomes (in particular level 1 outcomes).

2. Put more emphasis on designing recursive solutions.

3. Reserve more time for covering design methods and other problem-solving strategies, aside from labs.

4. Provide extra programming and testing tutorials to compensate with the students' programming weaknesses. In particular, tutorials could be provided in labs on simple concepts, or we could make use of PLTL sessions to review the unclear points of java implementation.

Comments by Committee:

Strengths of course:

- The majority of outcomes was met.
- Instruments test outcomes at the appropriate level.
- Emphasis on describing work done by means of a report is a strong point of lab assignments.

Opportunities for improvement:

Issues:

- Limited coverage of topics in graphs, in particular single-source shortest paths and topological sorting.
- Lack of separate assessment of understanding of specific graph algorithms.
- There was no redundancy in assessment, as graphs were only tested in final exam.

Possible approaches to address issues:

Include questions about graphs in third partial exam, as well as final.
Arrange to spend more time on graph algorithms – this may be difficult, as course covers a large amount of material.

Recommendations for change to course delivery:

The amount of guidance given for labs and specially lab reports could be progressively reduced as the semester progresses to encourage more initiative from students.

Recommendations for changes to program:

Outcomes, in particular those at level 3, are somewhat vague, thus it is hard to map instruments to them. It is suggested that the committee work on modifying them to simultaneously align better with ABET requirements and ease assessment.

The amount of material is excessive for a one-semester course. We suggest a few non-essential topics be eliminated, including: threaded trees, quadtrees, and edge list representations of graphs.

Assessment Instruments

CS2402 – Data Structures **NAME:.....**
MIDTERM 1
80 minutes – 60 points

This exam is to be done individually. It is a closed-notes closed-books exam.
Your copy has to be clean, your answers readable. All answers need to be justified.
Failing to meet any of the requirements will result in at least 5 points off.

Exercise 1. [5 points] Give the running time function and big-Oh of the following fragment of code:

```
for (int i=1; i<=n; i*=7)
    sum++;
```

Exercise 2. [10 points] Give the running time function and big-Oh of the following fragment of code:

```
for (int i=0; i<n; i+=7)
    for (int j=0; j<i; j+=2)
        sum++;
```

Exercise 3. [15 points] Give the running time function and big-Oh of the following fragment of code:

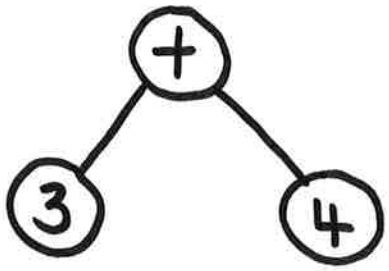
```
for (int i=1; i<=n; i*=7)
    for (int j=0; j<i; j+=2)
        sum++;
```

Exercise 4: [10 points] What is the time complexity of an algorithm whose time complexity can be expressed as the following recurrence formula:

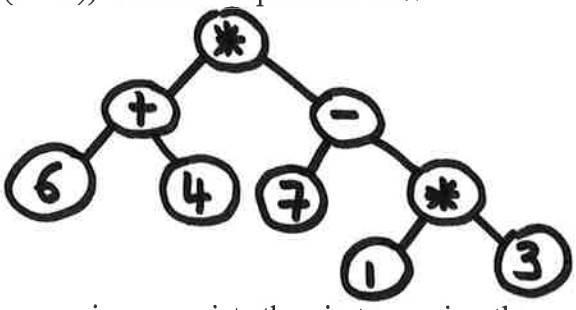
$F(n) = 2*F(n/2) + n$
$F(1) = 1$

Exercise 5. [20 points] Arithmetic expressions can be represented using what we call expression trees. For instance,

- 3 + 4 would be represented using the following tree:



- (6 + 4) * (7 – (1 * 3)) would be represented as:



A way of evaluating expressions consists then in traversing the corresponding expression tree appropriately. In particular, the way evaluations can be performed is by:

- first, evaluating the subtrees;
- second, combining the results obtained in each of the subtrees with the operator that is at the root.

Question 1 [10 points] Write the pseudo-code of the algorithm that performs such evaluations.
Note: your algorithm should not be recursive and has to make use of a stack.

Question 2 [5 points] As mentioned earlier in the description of the problem, your algorithm (your answer to Question 1) is a traversal. What kind of traversal is this?
Hint: this is one of the traversals / DFS-like algorithm that we covered in class.

Question 3 [5 points] Your algorithm (i.e., your answer to Question 1) is related to the method for expression evaluation that you read about in the textbook. How are these two algorithms related?

CS2402 – Data Structures **NAME:.....**
MIDTERM 2
80 minutes – 75 points (4 pages / 4 exercises)

This exam is to be done individually. It is a closed-notes closed-books exam.
Your copy has to be **clean**, your answers **readable**. **All answers need to be justified**.
Failing to meet any of the requirements will result in from 5 to 10 points off.

HITORI
Exercise 1 [10 points]:

- What is the time complexity of the simple Depth-First Search (backtracking) algorithm in the case of solving a Hitori puzzle of size n x n? Justify your answer. ***[5 points]***

- How to modify the DFS algorithm in the case of the Hitori, in order to improve this time complexity? Justify your answer. ***[5 points]***

DISCRETE EVENT SIMULATION
Exercise 2 [20 points] In the case of the bank simulation covered in class:

- What are the expected outcomes of the simulation? ***[5 points]***

- What are the events that are considered? Why not more? Why not less? ***[5 points]***

- Point out and describe in details two main differences that exist between the bank simulation seen in class and the drive-through simulation of lab3. ***[10 points]***

Difference 1:

Difference 2:

HASH TABLES
Exercise 3 [20 points]

- What is perfect hashing? ***[5 points]***

- Suppose that you need to hash social security numbers (SSNs). Propose a hashing function for this problem. Justify all the features of your hashing function. ***[5 points]***

- Now, suppose that you need to hash the 800 UTEP ID numbers. Propose a hashing function for this problem. Justify all the features of your hashing function. ***[5 points]***

- How is this second hashing function different from that of the SSNs? Justify the reason why you did not design them the same way. ***[5 points]***

TREES

Exercise 4 [25 points]

- Suppose that you are given a tree of p levels.
    - What is the configuration of such a tree with the minimum number of nodes? And what is this number of nodes? ***[5 points]***

CS2402 – Data Structures NAME:.....
MIDTERM 3
80 minutes – 95 points

This exam is to be done individually. It is a closed-notes closed-books exam.
Your copy has to be **clean**, your answers **readable**. **All answers need to be justified**.
Failing to meet any of the requirements will result in from 5 to 10 points off.

BINARY TREES
Exercise 1 [25 points]:

- Suppose that you have to implement a binary tree that has n levels, using an array.
  - What is the size of the array that you need to allocate. Justify your answer. ***[10 points]***

  - Explain how you use an array to represent a binary tree. In other words, given an array [a1, a2, a3, a4, a5, ... , ak], what is the corresponding binary tree? ***[5 points]***

- Suppose that you have to implement a complete binary tree with n elements.
  - What is the number of levels in the corresponding binary tree? ***[10 points]***

BINARY SEARCH TREES

Exercise 2 [15 points]

- Suppose that you have to use a binary search tree in order to find the kth largest element of a set of elements. ***[15 points]***
  - Describe the binary search tree that you use, with the possible enhancements that you want to bring to it.
  - Describe the algorithm for find-kth-largest(BinarySearchTree t, integer k) that takes as an input a binary search tree t (possibly enhanced as you will have described) and an integer k, and returns the kth largest element within tree t.

AVL TREES

Exercise 3 [30 points]

- What is an AVL tree? ***[5 points]***

- Build an AVL tree with the following sequence of numbers. Explain each step of the process, name each transformation, and keep the indices of the tree updated.

  9, 1, 7, 2, 5, 6, 8, 3, 4

  [10 points]

- Describe, using commented pseudocode, the algorithm for inserting a new node into an AVL tree. ***[10 points]***

- What is the cost of searching for a given element in an AVL tree? Justify your answer. ***[5 points]***

CS2402 – Data Structures **NAME:**.....

FINAL EXAM: Tuesday December, 9th, from 1pm to 3:45pm

165 minutes (estimated actual time: 90 to 105 minutes) – 100 points + 10 extra-credit

---

GRADE:

I: /25; II: /25; III: /35; IV: /25

This exam is to be done individually. It is a closed-notes closed-books exam.
Your copy has to be clean, your answers readable. All answers need to be justified.
Failing to meet any of the requirements will result in at least 5 points off. Besides, any answer that is not fully justified will only be given half of the full potential credit at the most.

PART I: Time complexity [25 points]

Exercise 1. [10 points] Give the running time function and big-Oh of the following fragment of code:

```
for (int i=0; i<n; i+=4)
    for (int j=0; j<i; j+=3)
        sum++;
```

Exercise 2: [10 points] What is the time complexity of an algorithm whose time complexity can be expressed as the following recurrence formula:

$$F(n) = 3*F(n/3) + n$$
$$F(1) = 1$$

Exercise 3: [5 points] Consider the algorithm of **bubble sort**. Let us denote the time complexity to run bubble sort on an input of size n by T(n). What is the recurrence formula that describes T(n) in terms of T of a smaller input size.

PART II: Stacks and queues [25 points]

Exercise 4: [5 points] How are a **queue** and a **priority queue** different? Similar?

Exercise 5: [20 points] Consider the following algorithm A:
Given a binary tree T to explore and an integer k as an input, algorithm A traverses and outputs T in a BFS (=level order) manner over the first k levels, and then performs a DFS (=pre-order) traversal of the remaining levels.

1. What is the result of running A with input (T,3)? where T is defined as follows: [5 points]

2. Write the commented pseudo-code of algorithm A. [15 points]

PART III: Trees [35 points]

Exercise 6: [10 points] Describe the following tree:

Exercise 7: [15 points] What is the cost of:

- Traversing a binary search tree of n elements?

- Searching for an element in an AVL tree of n elements?

- Searching for an element in a min heap of n elements?

Exercise 8: [10 points] Sort the following sequence of numbers in increasing order, using heap sort, as studied in class. Make clear the data structure(s) you use and show each step of your sorting.

9, 1, 8, 2, 7, 3, 6, 4, 5

PART IV: Graphs [25 points]

Exercise 9. [25 points] Consider the following graph, represented as an adjacency matrix:

0	1	0	1	1	0
0	0	0	0	1	0
0	1	0	0	0	1
0	0	0	0	0	0
0	0	1	1	0	1
0	0	0	0	0	0

1. What kind of graph is this? [5 points]

2. Is the representation of this graph as a matrix a good choice? [5 points]

3. Is this graph connected? Strongly connected? If not, how many connected or strongly connected components are there in this graph? [5 points]

4. If you were to implement an algorithm that:
   a. Detects if a graph of the same kind is connected;
   b. And if it is not, determines the number of (strongly) connected components;

   how would you proceed? Write the outline of such an algorithm. [10 points]

CS2402 – Data Structures
1st lab assignment
To be turned in on September, 11th at 11:59pm

In this assignment, you have to consider each of the following algorithms:

1. **Insertion sort**;
2. **Bubble sort**;
3. **Merge sort.**

For each of these algorithms, your assignment consists in:

1. **Theoretical analysis**:
   1. determining their running time function "on the paper" by using the recursive version of their implementation (hint: you have to use a recurrence formula);
   2. providing their big-Oh, worst-case and best-case time complexity;

2. **Experimental analysis:**
   1. designing a testing strategy to test each of the above-mentioned algorithms (their non-recursive version) to confirm the theoretical results obtained at step one of this assignment;
   2. running the corresponding experiments;
   3. reporting the results of your experiments, i.e., the number of steps performed for each algorithm for different input sizes (and [hint] for different input data of the same size);
   4. analyzing the results and discussing them.

You are expected to turn in:

1. **A report containing** (a template is available on the webpage of this course and also at the bottom of this lab subject):
   1. an introduction: describing the topic of this assignment and providing an outline of the report;
   2. a section per algorithm, each section containing the following subsections:
      1. a description of the algorithm: in plain english;
      2. the pseudocode of the algorithm: both the recursive and non-recursive versions;
      3. the theoretical analysis of the recursive version, along with the big-Oh, best and worst case;
      4. the experimental analysis where you first present your testing strategy, then you report the results of your experiments, and finally analyze and discuss them;
      5. a last subsection where you compare the theoretical and experimental results and you discuss their being similar or different;
   3. a conclusion: in which you recall what was done, what was learned from doing this lab, what were the challenges, how they were overcome.

2. **Your code containing**:
   1. the well-indented, well-documented java code of the above-mentioned algorithms (non-recursive version);
   2. note: it is important in your code to point out the counters that you use in order to keep track of the number of steps run within the algorithm.

How to submit:

To whom:
This assignment is to be turned in to **both** your TA, Jaime Nava, jenava@miners.utep.edu, and your professor, Martine Ceberio, mceberio@utep.edu.

Format of the email:
When submitting,the subject of your email **has to be**:

[cs2402] lab1: YourLastName YourFirstName.

For instance, when *Luis Garcia* submits his assignment, the subject of his email will be:

[cs2402] lab1: Garcia Luis.

Failing to do so will result in 5 points off.

Format and name of the submitted files:
The files that you have to attach to your email when submitting are:

- the file of **your report** in .doc or .pdf format (.docx not allowed): the name of your file has to be: YourLastName-YourFirstName.doc or YourLastName-YourFirstName.pdf
- the compressed file of **your code along with experiments** (if relevant) in .tar, .zip, or .tar.gz format (.rar not allowed): the name of your file has to be YourLastName-YourFirstName.tar or YourLastName-YourFirstName.zip, or YourLastName-YourFirstName.tar.gz.

Failing to follow the format will result in 5 points off per mistake.

Template for the report:

CS2402 -- LAB 1
YourLastName YourFirstName
Due date
---

1. Introduction
* describe the topic of this assignment
* provide an outline of the report

2. Insertion sort

2.1. Description of the algorithm
2.2. Pseudocode of the algorithm
Both the recursive and non-recursive versions
2.3. Theoretical analysis
Analysis of the recursive version of insertion sort
Big-Oh notation
Best and worst case: along with the corresponding configuration of the input data
2.4. Experimental analysis
* your testing strategy
* results of your experiments
*analysis and discussion of the results
2.5. Comparison

* you compare the theoretical and experimental results
* you discuss their similarities or differences

3. Bubble sort

3.1. Description of the algorithm
3.2. Pseudocode of the algorithm
Both the recursive and non-recursive versions
3.3. Theoretical analysis
Analysis of the recursive version of bubble sort
Big-Oh notation
Best and worst case: along with the corresponding configuration of the input data
3.4. Experimental analysis
* your testing strategy
* results of your experiments
*analysis and discussion of the results
3.5. Comparison
* you compare the theoretical and experimental results
* you discuss their similarities or differences

4. Merge sort

4.1. Description of the algorithm
4.2. Pseudocode of the algorithm
Both the recursive and non-recursive versions
4.3. Theoretical analysis
Analysis of the recursive version of merge sort
Big-Oh notation
Best and worst case: along with the corresponding configuration of the input data
4.4. Experimental analysis
* your testing strategy
* results of your experiments
*analysis and discussion of the results
4.5. Comparison
* you compare the theoretical and experimental results
* you discuss their similarities or differences

5. Conclusion

* recall what was done, what was learned from doing this lab, what were the challenges, how they were overcome

CS2402 – Data Structures
2nd lab assignment
To be turned in on October, 5th at 11:59pm

This assignment is intended to make you practice on

1. Stacks
2. Backtracking

Objectives:

1. Understand the backtracking algorithm and its power to generate and test possible paths to solution
2. Refresh your practice of stacks
3. Understand the notion of efficiency of programs through practice on a logic puzzle: Hitori

Description of the logic puzzle:

In the game of Hitori, you are given a 9x9 grid filled with numbers ranging from 1 to 9. The objective of the game is to shade squares (i.e., cells of your grid) so that numbers don't appear in a row or column more than once. More information at: http://www.menneske.no/hitori/eng/ and many others.
This puzzle can be solved using a backtracking procedure, using a stack.

You are expected to implement and program a solution to the Hitori puzzle.

You are expected to turn in:

1. **A report containing**:
   1. an introduction: describing the topic of this assignment and providing an outline of the report;
   2. a description of your algorithm (simple backtracking): in plain english; in particular, you have to justify/prove that your algorithm actually solves the Hitori puzzle;
   3. the pseudocode of the algorithm (not code!, and well commented);
   4. the experimental analysis of your algorithm, where you first present your testing strategy, then you report the results of your experiments, and finally analyze and discuss them;
   5. EXTRA CREDIT (15 points): a last subsection where you propose a possible improvement of your algorithm, implement it, and compare it through experimental results to your first version of it;
   6. a conclusion: in which you recall what was done, what was learned from doing this lab, what were the challenges, how they were overcome.

2. **Your code containing**:
   the well-indented, well-documented java code of your algorithm(s);

How to submit:

To whom:
This assignment is to be turned in to **both** your TA, Jaime Nava, jenava@miners.utep.edu, and your professor, Martine Ceberio, mceberio@utep.edu.

Format of the email:
When submitting,the subject of your email **has to be**:

[cs2402] lab2: YourLastName YourFirstName.

For instance, when *Luis Garcia* submits his assignment, the subject of his email will be:

[cs2402] lab2: Garcia Luis.

Failing to do so will result in 5 points off.

Format and name of the submitted files:

The files that you have to attach to your email when submitting are:

- the file of **your report** in .doc or .pdf format (.docx not allowed): the name of your file has to be: YourLastName-YourFirstName.doc or YourLastName-YourFirstName.pdf
- the compressed file of **your code along with experiments** (if relevant) in .tar, .zip, or .tar.gz format (.rar not allowed): the name of your file has to be YourLastName-YourFirstName.tar or YourLastName-YourFirstName.zip, or YourLastName-YourFirstName.tar.gz.

Failing to follow the format will result in 5 points off per mistake.

CS2402 – Data Structures
3rd lab assignment
To be turned in on October, 29th at 11:59pm

This assignment is intended to make you practice on
1. Discrete event simulation
2. The use of priority queues (note: not their implementation)

Objectives:
1. Understand simulation: why do we run simulation? What are the expected outcomes? How do we obtain them?
2. Understand that even a seemingly complex simulation can be implemented with few relevant events
3. Become proficient at manipulating queues and priority queues in event simulation.

Description of the situation to simulate:
For this lab, you will have to simulate a drive-through service (for instance, that serves coffee and other pastries☺). You have a certain number of waiters (this number will vary depending on your simulations because it is part of the expected outcome). You have customers coming at different times with different expected services (for which you know the expected preparation time).

Note: In this simulation, we will disregard the time it takes for a customer to order.

What information we seek out of this simulation:
Determine the best number of waiters, i.e, the number of helpers so that customers don't have to wait too long (and the queue of cars is not overwhelming for the parking lot) and the waiters are not idle too often and for too long. Of course, it is your responsibility to decide what "too long" means for the queue of cars, the idle time of waiters, as well as "too often".

You are expected to turn in:
1. **A report containing**:
   1. an introduction: describing the topic of this assignment and providing an outline of the report;
   2. a description of your algorithm: in plain english; in particular, you have to justify/prove that your algorithm does what it is supposed to do;
   3. the pseudocode of the algorithm (not its code, and it has to be well commented): here goes the management of the queues (and priority queues);
   4. the simulations you ran; where you first present your testing/simulation strategy, then you report the results of your experiments, and finally analyze and discuss them; the discussion should obviously lead to the outcome that was expected out of the simulation;
   5. a conclusion: in which you recall what was done, what was learned from doing this lab, what were the challenges, how they were overcome.
2. **Your code containing**:

the well-indented, well-documented java code of your algorithm(s);

How to submit:

To whom:
This assignment is to be turned in to **both** your TA, Jaime Nava, jenava@miners.utep.edu, and your professor, Martine Ceberio, mceberio@utep.edu.
Failing to do so will result in 5 points off.

Format of the email:
When submitting,the subject of your email **has to be**:
[cs2402] lab3: YourLastName YourFirstName.
For instance, when *Luis Garcia* submits his assignment, the subject of his email will be:
[cs2402] lab3: Garcia Luis.
Failing to do so will result in 5 points off.

Format and name of the submitted files:
The files that you have to attach to your email when submitting are:
- the file of **your report** in .doc or .pdf format (.docx not allowed): the name of your file has to be: YourLastName-YourFirstName.doc or YourLastName-YourFirstName.pdf
- the compressed file of **your code along with experiments** (if relevant) in .tar, .zip, or .tar.gz format (.rar not allowed): the name of your file has to be YourLastName-YourFirstName.tar or YourLastName-YourFirstName.zip, or YourLastName-YourFirstName.tar.gz.

Failing to follow the format will result in 5 points off per mistake.

CS2402 – Data Structures
4th lab assignment
To be turned in on November, 20th at 11:59pm

This assignment is intended to make you practice on

1. Trees: different kinds of trees;
2. Going back and forth from general trees to their binary representation.

Objectives:

1. Understand how to deal with a binary tree to represent a general tree.
2. Become proficient at manipulating trees
3. Get acquainted with DAGs.

Description of the situation to simulate:

In this lab, you have to build and manage **genealogical trees**. There are two different versions to be implemented (one for regular credit, the second one for extra credit).

For each version, you will have to implement the following functionalities:

- building a tree:
  file → tree
  + other methods depending on the version
- finding someone in a given tree:
  name x tree → boolean
- inserting someone in a given tree:
  name x tree → tree
  + other methods depending on the version
- determining relationships between two people:
  name x name x tree → type of relationship (if any)
- determining if two people are related:
  name x name x tree → boolean
- printing a tree:
  tree → void

Hereafter follow the specific features of each of the two versions:

1st version (compulsory): Genealogical trees that are such that:

Building genealogical trees can only result from:

1. Reading a text file containing the necessary information; e.g., lines of "Parent(x,y)" meaning that x is the parent of y. So your whole tree is based on this unique kind of information;
2. Adding new members in a tree by specifying:
   1. the tree T in which you want to add information

2. the node N that is going to receive a new child C

by invoking: T.add(N,C)

Such a tree is originally a general tree. You have to implement it as a binary tree (first-child next-sibling representation). In particular, the printing method that you have to implement has to print the original general tree, not the stored binary tree.

2nd version (extra credit: 15 points): Genealogical trees that are such that:

1. Reading a text file containing the necessary information; e.g., lines of "Parent(x,y)", meaning that x is the parent of y, or "Married(x,z)", meaning that x is married to z. So your whole tree is based on these only two kinds of information;
2. Adding new members in a tree by specifying:
   1. the tree T in which you want to add information
   2. the node N that is going to receive a new child C

   by invoking: T.addParent(N,C)
   3. the node N that is going to receive a new spouse S, which in turn belongs to its own genealogical tree T2:

   by invoking: addSpouse(N,T,S,T2)

   Such a method will result in merging part of T and T2.

Such a tree is originally a general directed acyclic graph (DAG): i.e., trees that are explicitly directed and can be connected together but without cycles, meaning that it can allow to "link" two trees when someone from one tree is married to someone from another tree.
In particular, when asked to print a given tree T, you will have to print it (as in the first version), including information about the spouses.
You will have to implement it as binary trees (first-child next-sibling representation).

You are expected to turn in:

1. **A report containing**:
   1. an introduction: describing the topic of this assignment and providing an outline of the report;
   2. a section per version, each containing:
      1. a description of your implementation(s) and algorithm(s): in plain english; in particular, you have to justify/prove that your algorithm does what it is supposed to do;
      2. the pseudocode of the algorithms (not its code, and it has to be well commented);
      3. the experiments you run to show that your implementation is sound and its performance practical.
   3. a conclusion: in which you recall what was done, what was learned from doing this lab, what were the challenges, how they were overcome.

2. **Your code containing**:
   the well-indented, well-documented java code of your algorithm(s);

How to submit:

To whom:
This assignment is to be turned in to **both** your TA, Jaime Nava, jenava@miners.utep.edu, and your professor, Martine Ceberio, mceberio@utep.edu.
Failing to do so will result in 5 points off.

Format of the email:
When submitting,the subject of your email **has to be**:

[cs2402] lab4: YourLastName YourFirstName.

For instance, when *Luis Garcia* submits his assignment, the subject of his email will be:

[cs2402] lab4: Garcia Luis.

Failing to do so will result in 5 points off.

Format and name of the submitted files:
The files that you have to attach to your email when submitting are:

- the file of **your report** in .doc or .pdf format (.docx not allowed): the name of your file has to be: YourLastName-YourFirstName.doc or YourLastName-YourFirstName.pdf
- the compressed file of **your code along with experiments** (if relevant) in .tar, .zip, or .tar.gz format (.rar not allowed): the name of your file has to be YourLastName-YourFirstName.tar or YourLastName-YourFirstName.zip, or YourLastName-YourFirstName.tar.gz.

Failing to follow the format will result in 5 points off per mistake.

CS2402 – Fall 2008
Lab assignment 5
Graphs
To be turned in on December, 8th at 11:59pm

Topics:

- Graphs
- Minimum Spanning Trees
- Shortest Path

Objectives:

- Become familiar with graphs: building, traversal
- Understand the minimum spanning trees and shortest path problems
- Become familiar with practical applications of graphs
- Understand the notion and importance of testing and how to report and analyze results

---

Description of the problem:

You are given the description of a network: computers (nodes / vertices), connections (edges), and the cost of each connection (weights). This network is described in terms of a matrix M of size nxn, where each element M[i,j] of the matrix is an integer that represents the cost of a connection from computer i to computer j. Let us note that M[i,j] can be different from M[j,i]. The graph described by this matrix is denoted G.

You have to determine the sub-network that connects all computers at minimum cost. This second network is denoted G'.

Once you have G and G', when given two computers (nodes / vertices) C1 and C2, you have to determine the cheapest connection path between C1 and C2 in G, and then in G', and compare the results.

You are expected to turn in:

1. **A report containing**:
    1. an introduction: describing the topic of this assignment and providing an outline of the report;
    2. a section per problem (1st: building the subnetwork / 2nd: finding the cheapest connections), each containing:
        1. a description of your implementation(s) and algorithm(s): in plain english; in particular, you have to justify/prove that your algorithm does what it is supposed to do;
        2. the pseudo-code of the algorithms (not its code, and it has to be well commented);
        3. the description of the performance of your algorithm;
        4. the experiments you run to show that your implementation is sound and its performance practical.

3. a conclusion: in which you recall what was done, what was learned from doing this lab, what were the challenges, how they were overcome.

2. **Your code containing**:
   the well-indented, well-documented java code of your algorithm(s);

How to submit:

To whom:
This assignment is to be turned in to **both** your TA, Jaime Nava, jenava@miners.utep.edu, and your professor, Martine Ceberio, mceberio@utep.edu.
Failing to do so will result in 5 points off.

Format of the email:
When submitting,the subject of your email **has to be**:

[cs2402] lab5: YourLastName YourFirstName.

For instance, when *Luis Garcia* submits his assignment, the subject of his email will be:

[cs2402] lab5: Garcia Luis.

Failing to do so will result in 5 points off.

Format and name of the submitted files:
The files that you have to attach to your email when submitting are:

1. the file of **your report** in .doc or .pdf format (.docx not allowed): the name of your file has to be: YourLastName-YourFirstName.doc or YourLastName-YourFirstName.pdf
2. the compressed file of **your code along with experiments** (if relevant) in .tar, .zip, or .tar.gz format (.rar not allowed): the name of your file has to be YourLastName-YourFirstName.tar or YourLastName-YourFirstName.zip, or YourLastName-YourFirstName.tar.gz.

Failing to follow the format will result in 5 points off per mistake.
