Sample Final Exam for CSCI 2410 (Java) Covers Chapters 18-29

FINAL EXAM AND COURSE OUTCOMES MATCHING

COURSE OUTCOMES

Upon successful completion of this course, students should be able to

1. design recursive solutions.
2. analyze algorithm complexities.
3. describe and analyze sorting algorithms.
4. use Java Collections Framework to develop applications.
5. implement classic data structures: array lists, linked lists, stacks, queues, heaps, binary trees, hash tables.
6. represent and solve problems using graph algorithms.

Here is a mapping of the final comprehensive exam against the course outcomes:

Question	Matches outcomes
Part I(1)	5
Part I(2)	5
Part I(3)	3
Part I(4)	6
Part I(5)	6
Part II(1)	1
Part II(2)	4
Part II(3)	2
Part II(4)	1, 5

Name:________________________	CSCI 2410 Data Structures and Algorithms
Covers Chapters 18-29	Armstrong Atlantic State University
Final Exam	Instructor: Dr. Y. Daniel Liang

Please note that the university policy prohibits giving the exam score by email. If you need to know your final exam score, come to see me during my office hours next semester.

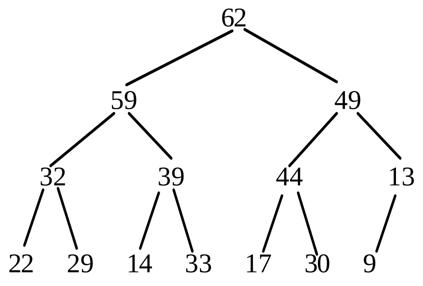
I pledge by honor that I will not discuss this exam with anyone until my instructor reviews the exam in the class.

Signed by ____________________ Date ____________________

Part I:

1. (2 pts) Add the elements 4, 15, 29, 101, 13, 2, into a heap in this order. Draw the final heap.

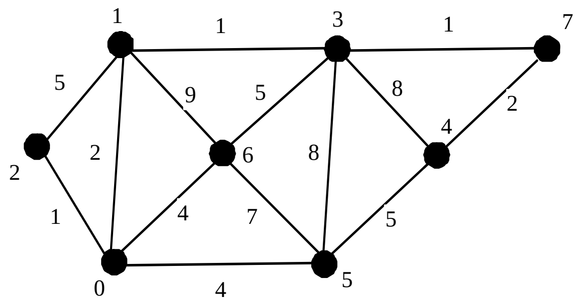
2. (2 pts) Given the following heap, show the resulting heap after removing 62 from the heap.



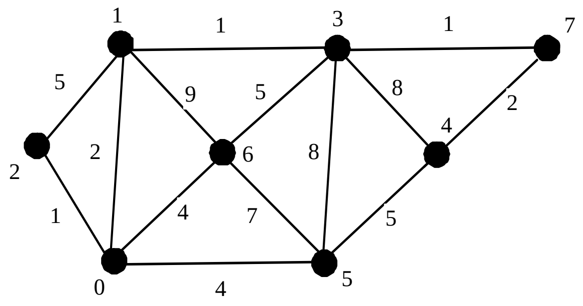
3. (2 pts) For the quick sort, show the partition of the following list using the first element as the pivot.

{45, 34, 342, 102, 3, 5, 35, 29, 244, 34}

4. (4 pts) Show the minimum spanning tree rooted at vertex 3 for the following graph using the algorithm in the book. Mark the order of the edges in which they are added into the minimum spanning tree. What is the total weight?



5. (4 pts) Show the shortest path tree rooted at vertex 3 for the following graph using the algorithm in the book. Mark the order of the edges in which they are added into the shortest path tree.



6 (3 pts)
Assume the load factor threshold is 75%. Show the hash table of size 13 after inserting entries with keys 14, 1, 27, 28, using linear probing. Show the hash table after removing 1.

Part II: Complete the following programs.

1. (10 pts) Write a **recursive** method that returns the number of the uppercase letters in a string using the following method header:

```
        public static int countUppercaseLetter(String s)
```

For example, countUppercaseLetter("ABc") returns 2.

2. (10 pts) Write a program that reads words from a text file and displays all the words (duplicates allowed) in ascending alphabetical order. The text file is passed as a command-line argument.

3. (10 pts) Add the following new method in the BST class.

```
/** Returns the number of nodes in this binary tree */
public int getNumberOfNodes()
```

Requirements:

1. Don’t use return size;

2. Write a recursive method that returns the number of nodes starting from the root.

4. (10 pts) Add the following new method in the UnweightedGraph class that returns the number of edges.

```
int getNumberOfEdges();
```

Implement it.

(Optional) 5. (10 pts) For the 24-point game, introduced in Programming Exercise 20.7, write a program to find out the success ratio for the 24-point game, i.e., number of picks with solutions / number of all possible picks.

Hints:

1. Assume the following method is given:

```
  /* Returns true if the four cards have a solution */
  public static boolean findSolution(int a, int b, int c, int d)
```

2. Create 52 cards as

```
    int[] deck = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
        1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
        1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
        1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13};
```

3. Write the code that produces all combinations of picking four numbers from the deck.

4. Count the ones that have no solutions.

6. (10 pts) Write a method to multiply two matrices and analyze its complexity. The header of the method is as follows:

```
public static double[][] multiplyMatrix(double[][] a, double[][] b)
```

To multiply matrix a by matrix b, the number of columns in a must be the same as the number of rows in b, and the two matrices must have elements of the same or compatible types. Let c be the result of the multiplication. Assume the column size of matrix a is n. Each element c_{ij} is $a_{i1} \times b_{1j} + a_{i2} \times b_{2j} + \ldots + a_{in} \times b_{nj}$. For example, for two 3 × 3 matrices a and b, c is

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix}$$

where $c_{ij} = a_{i1} \times b_{1j} + a_{i2} \times b_{2j} + a_{i3} \times b_{3j}$.

```
public static double[][] multiplyMatrix(double[][] a, double[][] b) {

   // Fill in the code here









}
```