Name:______________________________ **Student ID:**_______________

CSSAP 443: Final Exam

Past Final Exam Questions

Problem 1. Wanting to challenge your algorithm analysis skills further, your manager gave you the following array based oneThirdMergeSort():

```
oneThirdMergeSort(myArray, startIndex, endIndex)
        if (endIndex > startIndex)
                oneThirdIndex = (startIndex+endIndex) / 3
                oneThirdMergeSort(myArray, startIndex, oneThirdIndex)
                oneThirdMergeSort(myArray, oneThirdIndex+1, endIndex)
                combineArray(myArray, startIndex, oneThirdIndex, endIndex)
```

a. What is the recurrence equation for the run time of oneThirdMergeSort(). Please show the details of your derivation.

b. Show that the run time of oneThirdMergeSort() is $O(n \lg n)$.

Problem 2.

For each of the following statements indicate whether the statement is true or false. If the statement is correct, briefly state why. If the statement is wrong, explain why and correct it. One-sentence explanations should suffice.

a. Finding the best route from UW Bothell to downtown Seattle is an optimization problem.

b. The solution for $T(n) = 3T(\frac{n}{3}) + \theta(\sqrt{n})$ is $T(n) = \theta(n \lg n)$

Problem 3.
Wanting to impress the management with your knowledge in 2D searching, you have proposed a solution to the polygon-searching problem. We have a web-based searching system for locating *regions* on a *map*. In this system, each *region* is represented by a *Tpolygon* object. A *map* displays a large set of *Tpolygon* objcets. We want to design a search structure that supports efficient selection of polygons based on the user's input (a T2dBound). You proposed to build a QuadTree structure to support the interaction with the users. Once again, being a junior software engineer working for you, I have implemented the *Tpolygon* class as:

```
class Tpolygon {
    private:
        // this and that you don't care about
    public:
        void    BoundingBox(T2dBound &);
                        // returns the 2d bounding box of this polygon
        bool    IntersetcsBound(const T2dBound &);
                        // returns true if polygon intersects the input 2d bound
};
```

Recognizing each QuadTree node may contain more than one polygon, you have designed the TpolygonLink, TquadTreeNode, and TquadTree classes to look like:

```
class TpolygonLink {
    public:
        Tpolygon        *fPolygon;
        TpolygonLink    *fNext;
};

class TquadTreeNode {
    private:
        TpolygonLink    *fPolygons;
                            // list of polygons inside this leaf node
        T2dBound        fBound;   // 2d bound of the this quadTree node
        TquadTreeNode   *fChildren[4];
                            // children of this node

        TpolygonLink*   filterPolygonList(TpolygonLink *inputList);
                            // go through the input polygon list, inputList,
                            // one link at a time examine each Tpolygon
                            // and create/return a new list containing only those
                            // polygons that either within or touches
                            // the fBound of this node.

        void            SetPolygonPointer(TpolygonLink *list)
                            {fPolygons = list;}
    public:
        void            SearchStructure(const T2dBound &);
                            // searchStructure simply prints out the
                            // addresses of the polygons that are found.
};

class TquadTree {
    private:
        TquadTreeNode   *fRoot;
                            // root of the tree
    public:
        TquadTreeNode*  BuildTree(T2dBound&, int level, TpolygonLink *list);
        void            SearchStructure(const T2dBound &);
};
```

Taking the approach that:

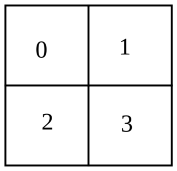
(i) Only leaf nodes will store polygons within (or touching) the 2d bounds of the node.
(ii) The QuadTree should not be more than *MaxHeight* level in depth. *Root is defined to be depth of 0.*
(iii) When (ii) is not violated, each leaf node should contain no more than *MaxPolygons* polygons.

Here is the implementation of TquadTree::BuildTree():

```
TquadTreeNode *TquadTree::BuildTree(T2dBound &bound, int level, TpolygonLink *list )
{
        TquadTreeNode  *node = new TquadTreeNode(bound);
                                        // the 4 fChildren[] and fPolygons are set to NULL
                                        // and fBound is set to the input bound

        TpolygonLink *newList = node->filterPolygonList(list);

        If ( WhatIsTheConditionHere ) {
                node->SetPolygonPointer(newList);
                return node;
        }

        T2dBound b[4];
        DefineTheBounds (bound, b) // for the 4 children.

        for (int i=0; i<4; i++)
                fChildren[i] = BuildTree(b[i], level+1, newList);

        return node;
}
```

a. Please show the pseudo code for ***WhatIsTheConditionHere***.

b. Given that the four children are defined according to:



Please show the results of ***DefineTheBounds()***, to compute (X_{min}, Y_{min}) and (X_{max}, Y_{max}) for the T2dBound for the four fChildren in TquadTree::BuildTree().

c. Please show the pseudo code for the implementation of the *SearchStructure(const 2dBound&)* methods for both TquadTree and TquadTreeNode. You can assume the existence of a function:

```
FindPolygonsInList(const T2dBound &bound, TpolygonLink*list)
                // go through the TpolygonLink list one link at a time
                // and print out the polygons in list that touch the bound.
```

d. Assuming polygons are very small when comparing to the size of the map (*can almost treat them as points)*, and we never reach the maximum level of *MaxHeight* when building the QuadTree, and that the polygons are uniformly distributed. What is the average runtime of your *TquadTreeNode::SearchStructure(const T2dBound&)* implementation, given that the user's input bound is *so small that you can treat it as a point*? Please express your answer as a function of total number of input polygons (***n***), and the ***MaxPolygons*** parameter defined in this problem.

Name:______________________________ **Student ID:________________**

Problem 4.
After CSSAP 443, you decided to take a vacation to cool down the over worked brain. You are faced with the problem that you have too many belongings and not sure of what to bring with you. Practicing problem-solving approaches we have learned, you organized your belongings into n bags. You then weigh each bag and assign a value according to how important each specific bag is. Your bags are $<b_1, b_2, ... b_n>$, the corresponding weight and value for each bag are stored in the **WeightArray =** $<w_1, w_2, ...w_n>$ and the **ValueArray =** $<v_1, v_2, ... v_n>$ accordingly. With a carrying capacity of **MaxWeight**, you want to maximize the total value from the bags you can carry. Flipping through your lecture notes, you realize you are just like the thief in the **0-1 Knapsack-Problem**. Reading the lecture notes further, you realize you can let *c[i,w]* be the maximum value you could carry if you have *i-bags* to choose from, with a maximum *carrying capacity of w*. Your worn-out lecture notes says here is how to solve for *c[n,MaxWeight]:*

```
for i = 0 to n
    c[i,0] = 0

for w = 0 to MaxWeight
        c[0, w] = 0

for i = 1 to n
    for w = 1 to MaxWeight
            if   (WeightArray[i] > w)
                    c[i,w] = WhatShouldThisBe
            else
                    value1 = DoComputeValue1
                    value2 = DoComputeValue2


                    if   (value1 > value2)
                            c[i,w] = value1
                    else
                            c[i,w] = value2
```

a. Please fill in ***WhatShouldThisBe***, ***DoComputeValue1***, and ***DoComputeValue2***.

Name:______________________________ **Student ID:________________**

Suppose you have 4 bags with:

WeightArray	**=**	<2, 4, 3, 5>	and the corresponding
ValueArray	=	<8, 10, 5, 7>	

After a Quarter of intense programming, algorithm analysis and lack of exercise, you realized you could only carry **MaxWeight** of 7.

b. How many rows and columns do you need in the c[i,w] table (on the next page) to compute the maximum values you could carry with you? Please explain your answer.

Weight

Objects

c. Please fill in the above c[i,w] table according to the given algorithm.

d. Based on the above table:

(i) What is the maximum value you will be taking with you?

(ii) Which are the bags you will be carrying?

(iii) The actual weight you will be carrying?