# STAT/BIOSTAT 534 Statistical Computing
# Spring Quarter 2015
# Final Project

Adrian Dobra
`adobra@uw.edu`

**The final project is due on Tuesday, June 9, 2015 at 11:00pm. You should use the dropbox to submit your code. Please note that you will be graded not only on how your code works, but also on how readable your code is.**

## Problem 1 (75 points)

You need to implement in C the following sorting algorithm for a vector `double v[n]`. Construct a binary tree corresponding with the vector `v` as we did in class. For each element `v[i]`, the left subtree should contain elements of `v` smaller than `v[i]`, while the right subtree should contain elements of `v` greater or equal to `v[i]`. Please use the code I gave you and implement only the parts of the code you do not already have.

Repeat the following steps until the tree becomes empty:

1. Find the node in the tree containing the smallest key (number).
2. Record this number.
3. Remove this node from the tree.

The resulting sequence of numbers will be the elements of `v` in increasing order. This sequence should be the output of your code. Your program should function properly for the file "somenumbers.txt".

## Problem 2 (75 points)

Please refer to the source code from our lecture in graphs. You need to write a C function

```
void findConComp(int myvertex,int** graph,int nvertices);
```

that prints out all the vertices that belong to the same component as the vertex "myvertex". Your "main.cpp" file should be:

```
int main()
{
   char graphfile[] = "data.txt";
   char dotfile[] = "data.dot";

   int nvertices = -1;
   int** graph = readgraph(graphfile,nvertices);

   //print out the graph in Graphviz format
   printgraph(dotfile,graph,nvertices);

   //find the connected component of vertex 1
   findConComp(0,graph,nvertices);
   //find the connected component of vertex 2
   findConComp(1,graph,nvertices);
   //find the connected component of vertex 5
   findConComp(4,graph,nvertices);

   //free memory
   freegraph(graph,nvertices);

   return(1);
}
```

## Problem 3 (150 points)

Write a C program that uses the Master/Slave scheme we studied in class to identify, for each vertex of a graph, the other vertices in its connected component. The Master process should proceed as follows:

1. The Master should ask each Slave process to identify the connected components of a vertex.
2. The Master should wait until each Slave process finishes its work and record the corresponding results. If there are any vertices whose components have not been identified, the Master should ask the Slaves to find their connected components.
3. The Master should output the the connected components of the input graph. Each connected component should be written only once in the output.

Each Slave process should proceed as follows:

1. The Slave should listen for a work order from the Master.

2. Once an order has been received, the Slave should determine the connected component for the vertex in the work order.
3. The Slave should transmit the connected component it determined back to the Master process.

Your code should run properly for the graph in the file ”data.txt”.