# CSCE 221 Assignment 4 Cover Page

First Name Last Name UIN

User Name E-mail address

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more on Aggie Honor System Office website: `http://aggiehonor.tamu.edu/`

| Type of sources | | | | |
|---|---|---|---|---|
| People | | | | |
| Web pages (provide URL) | | | | |
| Printed material | | | | |
| Other Sources | | | | |

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.
*On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.*

Your Name Date

# Assignment 4 (100 pts)

**Program: Due March 24 at 11:59 pm**

**Hardcopy Report: Return to your TA in lab**

**Programming (70 points)**

1. Write a C++ program that creates a binary search tree, empirically calculates the average search cost for each node in such a tree, and removes a designated node. Your program should:

   - Read integer data from the file designated by the user. Every line of the file contains one input integer. Assume that each integer in the input data file is unique. (Note that there is no number that gives the total number of integers in the file)
   - Print the input data on the screen.
   - Create a binary search tree based on the input data.
   - Calculate the search cost for each node **when it is inserted** and store it
     - A binary node element has at least two data fields: `Key` and `SearchCost`
     - For each node, count and store the number of comparisons required when searching for the node (i.e., the number of comparisons = the search cost for the node = 1+ the depth of the node)
   - Print the binary search tree by performing a traversal: If the number of nodes is less than $2^4$, print on the screen the resulting tree along with the information about the nodes (for each node, print both its key and search cost). Otherwise, print this tree and information about its nodes into a file. See the example below.
   - Print the binary search tree level-by-level: If number of nodes is less than or equal to $2^4$, print the tree level-by-level to a file. For trees larger than $2^4$, you do not need to output it as the file may be large. See the experiment section.
   - Remove a node designated by a user from the binary search tree and print the resulting tree on the screen if the number of nodes is less than $2^4$. **You should update the search costs of the tree nodes if the costs have been changed upon node removal.**
   - Calculate the average search cost for all nodes in the tree and print it
     - Sum up the search cost over all the nodes in your tree as you traverse the tree in one of three traversals
     - divide the sum by the total number of nodes in the tree, and the result is the average search cost for all the tree nodes
   - Print total number of nodes

2. Example:

   Input data:

   ```
   5
   3
   9
   7
   10
   11
   ```

   Create a binary search tree:

   ```
   Key = 5  SearchCost = 1
   ```

```
Key = 3  SearchCost = 2
Key = 9  SearchCost = 2
Key = 7  SearchCost = 3
Key = 10  SearchCost = 3
Key = 11  SearchCost = 4
Total number of nodes is 6
```

To generate output on the screen or to a file we use the in-order traversal. Here each node is represented as Key[SearchCost]:

```
3[2] 5[1] 7[3] 9[2] 10[3] 11[4]
```

Sum of the search cost over all the nodes in the tree is: $2+1+3+2+3+4=15$. Average search cost: $15/6=2.5$.

```
Average search cost is 2.5
```

Output the tree level-by-level to a file:

```
5
3 9
X X 7 10
X X X X X X X 11
```

Prompt users to remove a node by key. Replace the removed node by the minimum element in the right subtree, or the left child:

```
Please enter the key to remove:  9
3[2] 5[1] 7[3] 10[2] 11[3]
```

3. Download files containing integer data from the course website.

   (a) The files *1p~~.dat~~*, *2p~~.dat~~*, ..., *12p~~.dat~~* contain $2^1-1$, $2^2-1$,..., and $2^{12}-1$ integers respectively. The integers make 12 **perfect binary trees** where all leaves are at the same depth. Calculate and record the average search cost for each perfect binary tree.

   (b) The files *1r~~.dat~~*, *2r~~.dat~~*, ..., *12r~~.dat~~* contain same number of integers as above. The integers are randomly ordered and make 12 **random binary trees**. Calculate and record the average search cost for each tree.

   (c) The files *1l~~.dat~~*, *2l~~.dat~~*, ..., *12l~~.dat~~* contain same number of integers as above. The integers are in increasing order and make 12 **linear binary trees**. Calculate and record the average search cost for each tree.

   (d) Make a table and a plot showing the average search cost (y-axis) versus the number of tree nodes (x-axis) of all perfect binary trees, random binary trees and linear binary trees.

   (e) Provide the output in the text format for file *1p-4p*, *1r-4r*, and *1l-4l*. The nodes will be printed according to their depth level. Missing nodes will be represented by the symbol `X`. A possible solution is to fill the missing nodes with fake nodes in the data structure when printing the tree.

   EXAMPLE:

   ```
   5
   3 9
   X X 7 10
   X X X X X X X 11
   ```

4. **Extra credit.**

   (a) (10 points) Use the in-order traversal and any graphic programming library like FLTK for binary tree drawing. See the textbook, page 301 for the algorithm.

(b) (10 points) Implement a balanced binary search tree using one of the balancing technique like AVL, Red-Black or 2-4 tree.

5. Hints

   (a) Besides using links/pointers to represent a binary search tree, you may store the binary tree in a vector. This implementation might be easier, especially for the printing of a tree level by level. Please refer to section 7.3.5 at p.295 of the book.

   (b) To output a binary search tree level-by-level, you may use breadth-first-search (BFS) algorithm if you use link-based implementation.

      i. You may use the doubly linked list class from the assignment 3, or use the STL queue: http://www.cplusplus.com/reference/queue/queue/

      ii. The following is a sample pseudo-code of the BFS algorithm

```
Algorithm OutputTreeLevelByLevel(BinarySearchTree T)
   Queue q
   q.enqueue(T.root)
   while (not q.empty()) do
     TreeNode node = q.dequeue()
     /* output a node (you need to determine whether to output a newline) */
     print node.key
     if (there are still unenqueued non-NULL nodes in the tree) then
       /* enqueue children here for later output */
     endif
   done
```

**Report (30 points)**

Write a brief report that includes the following:

1. A description of the assignment objective, how to compile and run your programs, and an explanation of your program structure (i.e. a description of the classes you use, the relationship between the classes, and the functions or classes in addition to those in the lecture notes).

2. A brief description of the data structure you create (i.e., a theoretical definition of the data structure and the actual data arrangement in the classes).

3. A description of how you implement the calculation of (a) individual search cost and (b) average search cost and (c) updated search cost. Is the implementation associated with the operations find, insert, remove? Analyze the time complexity of (a) calculating individual search cost, (c) **updating** the search cost of an individual node and (b) **summing up** the search costs over all the nodes.

4. Give individual search cost in terms of $n$ using big-O notation. Analyze and give the average search costs of a perfect binary tree and a linear binary tree using big-O notation, assuming that the following formulas are true ($n$ denotes the total number of integers).

   $\sum_{d=0}^{\log_2(n+1)-1} 2^d(d+1) \simeq (n+1)\cdot\log_2(n+1) - n$ and $\sum_{d=1}^{n} d \simeq n(n+1)/2$

5. Include a table and a plot of average search costs you obtain. In your discussions of the experimental results, compare the curves of search costs with your theoretical analysis results in item 4.