The Hong Kong University of Science & Technology
Department of Computer Science

COMP 2012H: Honors OOP and Data Structures
Written Assignment
Due by 5pm, Monday, 30 November, 2015

Your answers will be graded on clarity, correctness, efficiency, and precision.

1. "Data Structures and Algorithms in C++," Goodrich et al, C-4.9
   Describe how to efficiently implement the queue ADT using two stacks.

2. "Data Structures and Algorithms in C++," Goodrich et al, C-4.18
   Describe an algorithm on how to implement stack ADT of `push` and `pop` using two queues labeled as Q_1 and Q_2.

3. "Data Structures and Algorithms in C++," Goodrich et al, C-4.3
   Suppose you are given an n-element array A containing distinct integers that are listed in increasing order. Given a number k, describe a recursive algorithm which returns true if there are two integers in A that sum to k, false otherwise.

4. Consider the following recursive program:

```
int fun(int A[], int p, int r ){

  if (p == r)
    return r;
  else {
    int m = (p+r)/2;
    int i = fun(A, p, m);
    int j = fun(A, m+1, r);

    if (A[i] >= A[j])
      return i;
    else return j;
  }
}
```

   (a) Suppose $A = \{5, 6, 12, 2, 4, 8, 16, 21\}$. What is the final output of `fun(A, 0, 7)` ?

   (b) Suppose $A = \{1, 1, 1, 1, 1, 1, 1, 1\}$. What is the final output of `fun(A, 0, 7)` ?

   (c) What does the program do or return?

5. Consider the following recursive procedure:

```
// n is any positive integer
// m is between 2 and 9
void foo( int n, int m ){
```

```
  if( n < m )
    cout << n;
  else{
    foo( n/m, m );
    cout << n%m;
  }
}
```

(a) What is the output for `foo(10, 2)`?

(b) What is the output for `foo(34, 5)`?

(c) What does the procedure do?

6. Consider the standard arithmetic expressions (infix expressions) with the following characteristics:

   - It is a legal infix expression with at most 100 characters
   - It has FOUR operators: exponentiation (`^`), multiplication (`*`), addition (`+`), subtraction (`-`), and possibly parentheses `(` and `)`
   - It has variables taken from the set $\{a, b, \ldots, z\}$
   - It contains no blanks

   Describe an algorithm to convert an infix expression into postfix expressions using a stack. You can assume that a stack operations are provided, so further explanation and description of these operations is not needed.

   Hint: Note that the four operators can be categorized into three groups according to their associativity. Let A, B, C and D be expressions that are either single variables or parenthesized expressions.

   - Group 1 consists only of the subtraction operator since it is the only operator that is left associative and not right associative. That is, $A - B - C$ can only be evaluated as $(A - B) - C$. Evaluating it as $A - (B - C)$ is incorrect.
   - Group 2 consists only of the exponentiation operator since it is the only operator that is right associative and not left associative. That is, $A\hat{}B\hat{}C\hat{}D$ can only be evaluated as $A^{(B^{(C^D)})}$.
   - Group 3 consists of the multiplication and addition operators since they are both left and right associative.

7. "Data Structures and Algorithms in C++," Goodrich et al, R-6.8
   Let T be a binary tree with more than one node.

   (a) Is it possible that the preorder traversal of T visits the nodes in the same order as the postorder traversal of T? If so, give an example; otherwise, argue why this cannot occur.

   (b) Is it possible that the preorder traversal of T visits the nodes in the reverse order of the postorder traversal of T? If so, give an example; otherwise, argue why this cannot occur.

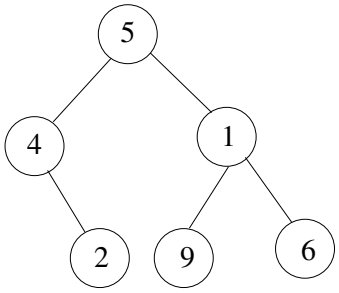

Figure 1: A binary tree.

8. This question is about the relations of preorder, inorder, and postorder traversals of binary trees with distinct elements stored in the nodes (i.e., distinct node labels).

   (a) Give the *preorder* traversal, *inorder* traversal, and *postorder* traversal of the binary tree with the indicated node labels as shown in Figure 1.

      i. Preorder:

      ii. Inorder:

      iii. Postorder:

   (b) You are given the preorder and inorder traversals of a binary tree with node labels as below:
   Preorder: 8, 6, 3, 4, 5, 1, 2;
   Inorder: 3, 6, 4, 8, 1, 5, 2.

      i. Draw the binary tree.
      ii. Give the corresponding postorder traversal of this tree.

   (c) Given the preorder and inorder traversals of a binary tree with distinct node labels, the corresponding binary tree can be uniquely determined. Describe an algorithm to determine that.
   Hint: You may answer this in two ways: 1) For any node in the tree, determine its left and right children; or 2) Give the postorder representation of the binary tree.

   (d) Actually it is known that given the postorder and inorder traversals of a binary tree with distinct node labels, the corresponding binary tree can also be uniquely determined. However, if only the preorder and postorder traversals are given, then the corresponding binary tree may not be uniquely determined.
   For example, let's say we have the preorder and postorder traversals of a binary tree as below:
   Preorder: 3, 4, 2, 1, 5;
   Postorder: 4, 5, 1, 2, 3.
   Give two distinct binary trees which yield the same traversal results as above.

9. "Data Structures and Algorithms in C++", Goodrich et al, C-9.7 & C-9.8
   Let D be a binary search search tree with n items.

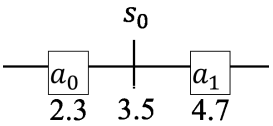

Figure 2: Two numbers a_0 and a_1 and their split value s_0 on a line.

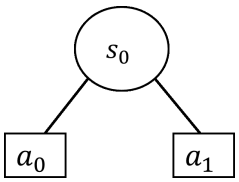

Figure 3: The perfect BST constructed based on $a_0 = 2.3$, $a_1 = 4.7$ and the split value $s_0 = 3.5$.

Describe an algorithm, `findAllInRange( k1, k2 )`, which returns all the elements with key k in D, where $k_1 \leq k \leq k_2$.

10. Suppose you are given N distinct real numbers. Given a value x, could you find the number closest to it, i.e., x's *closest neighbor*? Ties are resolved arbitrarily.

    For example, given two numbers 2.3 and 4.7. If $x = 3$, its closest neighbor is 2.3. If $x = 1.2$, the closest neighbor is 2.3. On the other hand, the closest neighbor for $x = 5.1$ is 4.7.

    Clearly, one can do a linear search to find the closest neighbor. In this problem, you are going to arrange the numbers as part of a binary search tree (BST) in order to efficiently support frequent queries on closest neighbors. Your search will runs much faster than linear time (runs in $O(\log N)$ time in the worst case).

    For simplicity, let's first consider $N = 2^k$ in this problem, where k is a positive integer. To start, first sort the numbers and labeled the sorted numbers from $a_0, a_1, \ldots, a_{N-1}$. Let's consider the simple case of $N = 2$, and you will generalize the case to arbitrary N in the following questions.

    Consider our example again with values $a_0 = 2.3$ and $a_1 = 4.7$. We first put them in a linear line according to Figure 2, with the midpoint at $s_0 = 3.5$. The midpoint s_0 is called the *split value*. It has significance that if a query is lower than s_0, then a_0 is the closest neighbor; otherwise a_1 is the closest neighbor.

    Given the above observation, one can form a *perfect* BST (i.e., a balanced BST with each level full) consisting of a_0, a_1 at the leaves (square nodes) and s_0 (circle node) as shown in Figure 3. The split value s_0 is the internal node (root in this case) of the tree.

    Given such a BST and a query x, one can simply traverse down the tree to search for x. The leaf node is the closest neighbor. For example, if $x = 3$, by searching for x we reach the leaf 2.3, which is the correct closest neighbor. On the other hand, if $x = 5.1$, we reach the leaf 4.7, which is also the correct closest neighbor.

    (a) Suppose now $k = 2$, i.e., $N = 4$ with $a[] = \{1.2, 2.5, 4.2, 5.7\}$ as shown in Figure 4.

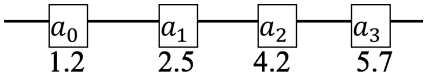

Figure 4: Four sorted numbers.

i. Please show below the perfect BST formed for this case with all the a_i's at the leaves (indicated as square nodes) and s_i (the split values) as internal nodes (indicated as circle) to support efficient closest neighbor queries.

ii. Given a value $x = 3.2$ and the constructed BST in Part 10(a)i, describe how you would find its closest neighbor.

(b) Suppose you are given a sorted list of $2^k - 1$ distinct real numbers labeled as $b_0, b_1, \ldots, b_{2^k-2}$.

i. Describe an efficient algorithm to put these numbers into a *perfect* binary search tree.

ii. In the tree you construct above, which b_is are the leave nodes?

(c) Consider the general case that $N = 2^k$. Given distinct sorted real numbers labeled $a_0, a_1, \ldots, a_{N-1}$, we can find their split values $s_0, s_1, \ldots, s_{N-2}$. A perfect BST can then be formed based on these $2N - 1$ numbers to support efficient closest neighbor queries.

i. Express s_i in terms of a_i and a_{i+1}, for $0 \leq i \leq N - 2$.

ii. Describe an algorithm to construct a perfect BST to support fast search on closest neighbor.

(d) Given an array `a[0...N-1]` of $N = 2^k$ sorted distinct real numbers, you want to output in sorted order all the a_i's within the range (x, y). For example, consider the case of Part 10a. Given a range of $(2, 5)$, you should output 2.5 and 4.2.

Using the perfect BST you obtained in Part 10c, or otherwise, describe an efficient algorithm on how to use it to achieve the above.

(e) Finally, consider now an arbitrary N such that $2^{k-1} < N < 2^k$. Describe how you would construct a BST which supports efficient search on closest neighbor.

Using the tree, describe how to search for the closest neighbor given a value x.

11. Consider inserting the keys 31, 20, 54, 3, 39, 28, 17, 78, 53, 62, 124, 22, 27, 41 into a hash table of length $m = 17$ using open addressing with the primary hash function $h_1(k) = k \bmod m$.

By drawing the hash table after inserting each key, illustrate the result of inserting these keys using

(a) linear probing; and

(b) double hashing, where the secondary hashing function is given by $h_2(k) = 1 + (k \bmod 13)$.