CSE 101
Spring 2026
Midterm Exam 2 Solutions

1. (20 Points) Using the List ADT from pa6, write a C++ client function with the heading

```
List SymmetricDifference(List A, List B)
```

that returns a new List representing the symmetric difference of the two input Lists A and B. In other words, the returned List will contain all elements that are either in A or in B, but not in both. These elements will occur in the same order that they occur in A, and B, and with the same frequencies. Thus compute everything in A not in B, followed by everthing in B not in A. The following examples illustrate the operation.

$A = (1, \mathbf{3}, 2, \mathbf{3}, 4, 1), B = (\mathbf{5}, 2, 4, \mathbf{6}, 1, 2) \longrightarrow$ return $(3, 3, 5, 6)$
$A = (\mathbf{4}, \mathbf{3}, 2, \mathbf{4}), B = (\mathbf{1}, 2, \mathbf{1}, \mathbf{1}) \longrightarrow$ return $(4, 3, 4, 1, 1, 1)$
$A = (1, 1, 2, 1), B = (2, 2, 1, 2) \longrightarrow$ return $(\,)$

Solution:

```
List SymmetricDifference(List A, List B){

   ListElement x;
   List L;

   // Everything in A not in B
   A.moveFront();
   while( A.position()<A.length() ){
      x = A.moveNext();
      B.moveFront();
      if( B.findNext(x) == -1 ){  // B does not contain x
         L.insertBefore(x);  // append x
      }
   }

   // Everything in B not in A
   B.moveFront();
   while( B.position()<B.length() ){
      x = B.moveNext();
      A.moveFront();
      if( A.findNext(x) == -1 ){  // A does not contain x
         L.insertBefore(x);  // append x
      }
   }

   return L;
}
```

■

2. (20 Points) Using the List ADT from pa6, write a C++ client function with the heading

```
void Remove(List& L, ListElement x)
```

that removes all occurances of x from L. All other elements of L will remain in their original order, and frequencies. If L does not contain x, then no changes to L will be made. For instance, if $L = (2, 3, 7, 3, 9, 3, 7, 1)$, then the call $\text{Remove}(L, 3)$, results in $L = (2, 7, 9, 7, 1)$.

Solution1:

```
void Remove(List& L, ListElement x){

   L.moveFront();
   while( L.findNext(x) >= 0 ){
      L.eraseBefore();
   }

}
```

■

Solution2:

```
void Remove(List& L, ListElement x){

   for( L.moveFront(); L.findNext(x)>=0; ){
      L.eraseBefore();
   }

}
```

■

Solution3:

```
void Remove(List& L, ListElement x){

   for( L.moveBack(); L.findPrev(x)>=0; ){
      L.eraseAfter();
   }

}
```

■

3. (20 Points) Use limits to determine whether the following statements are true or false. Justify your answers.

   a. (5 Points) $\sqrt[3]{n} = o\left(\sqrt{n}\right)$ **True**

   Solution:

$$\lim_{n\to\infty} \frac{\sqrt[3]{n}}{\sqrt{n}} = \lim_{n\to\infty} \frac{n^{1/3}}{n^{1/2}} = \lim_{n\to\infty} n^{1/3-1/2} = \lim_{n\to\infty} n^{-1/6} = 0$$

   Therefore $\sqrt[3]{n} = o\left(\sqrt{n}\right)$. ■

   b. (5 Points) $n\ln(n) = \omega(n)$ **True**

   Solution:

$$\lim_{n\to\infty} \frac{n\ln(n)}{n} = \lim_{n\to\infty} \ln(n) = \infty$$

   Therefore $n\ln(n) = \omega(n)$. ■

   c. (5 Points) $2(n-5)^3 = \Theta(n^3)$ **True**

   Solution:

$$\lim_{n\to\infty} \frac{2(n-5)^3}{n^3} = \lim_{n\to\infty} 2\left(1-\frac{5}{n}\right)^3 = 2$$

   Therefore $2(n-5)^3 = \Theta(n^3)$. ■

   d. (5 Points) If $0 < a < b$, then $b^n = \Omega(a^n)$. **True**

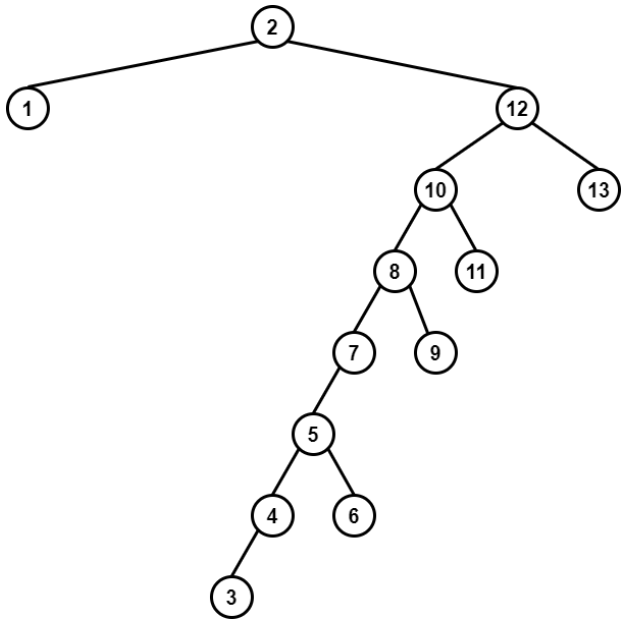
   Solution:

$$\lim_{n\to\infty} \frac{b^n}{a^n} = \lim_{n\to\infty} \left(\frac{b}{a}\right)^n = \infty \qquad \left(\text{since } \frac{b}{a} > 1\right)$$

   Therefore $b^n = \omega(a^n)$, and hence $b^n = \Omega(a^n)$. ■

4. (20 Points) Let T be a Binary Search Tree containing the keys $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13\}$. Suppose that a **pre-order tree walk** prints the keys in order: $2, 1, 12, 10, 8, 7, 5, 4, 3, 6, 9, 11, 13$, and that a post-order tree walk prints the keys in order: $1, 3, 4, 6, 5, 7, 9, 8, 11, 10, 13, 12, 2$. Determine the structure of T. (Note: only one of the two tree walks is really necessary since each of them uniquely determines the structure of T.) Present your solution by drawing a picture.

Solution:

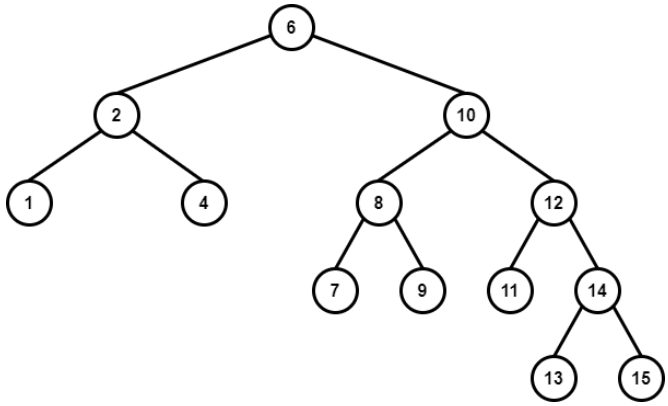


■

5. (20 Points) Use the `TreeInsert()` algorithm to insert the following keys: 6, 2, 1, 4, 10, 8, 7, 9, 12, 11, 14, 13, 15 (in order) into an initially empty BST.

   a. (10 Points) Draw the resulting BST

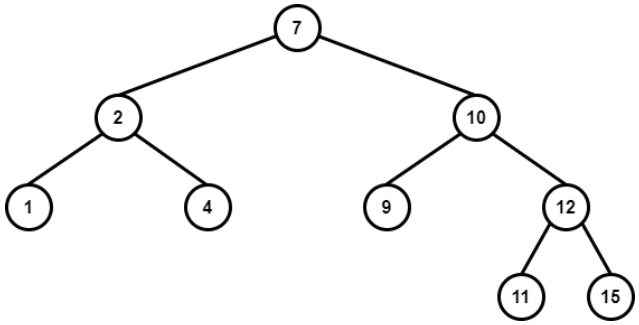
   Solution:



■

   b. (10 Points) Use the `Delete()` algorithm to delete the following keys: 8, 6, 13, 14 (in order) from the BST you drew in part (a), then draw the resulting tree.

   Solution:



■