1. (5%) Select five most essential operations to implement a queue data type.
   a. `bool empty()`: check whether the queue is empty
   b. `object front()`: return the value of the item at the front of the queue
   c. `object back()`: return the value of the item at the back of the queue
   d. `void popfront()`: remove the item from the front of the queue
   e. `void popback()`: remove the item from the back of the queue
   f. `void pushfront(object item)`: insert the argument item at the front of the queue
   g. `void pushback(object item)`: insert the argument item at the back of the queue
   h. `int size()`: return the number of elements in the queue

2. (5%) Use the binary search to find a target value in a list of 10,000 items. What is the maximum number of comparisons the search will require?

3. (15%) A complete binary tree B, containing 1,000 nodes (with indices 1, 2,..., 1000), is stored in an array. Answer the following questions.
   a. What is the level of the tree?
   b. How many nodes are leaf nodes?
   c. What is the index of the parent of B[215]?
   d. What is the index of the first node with one child?
   e. What is the range of the indices for all nodes at level 7 in the tree?

4. (5%) Draw a 2-3-4 tree that contains the following integers:
   
   2 4 5 8 10 13 15 24 35 38

5. (5%) Name all sequences that are possible inorder scans of some binary search trees.
   a. 67 8 2 19 4 5 11
   b. 2 91
   c. 39 20 18
   d. 1 3 5 4 2
   e. 44 31 12 10
   f. 59
   g. 4 5 10 11 12
6. (5%) Assume that we obtain the following integer sequence by traversing a binary search tree in postorder sequence:

   25 17 10 41 29 49 56 67 65 45

Draw a binary search tree that generates such an ordering.

7. (5%) Implement a recursive algorithm to count the number of interior nodes in a binary tree, each node having at most two children pointed by left and right.
   An interior node is a node having at least one child.

   template <typename T>
   void countNode(tnode<T>* t, int& count);

8. (5%) If a hash table with $n$ elements uses chaining with separate lists, name all correct statements regarding complexities for a search.
   a. Worst case for a successful search: $O(n)$
   b. Worst case for an unsuccessful search: $O(n)$
   c. Average case for a successful search: $O(1)$
   d. Average case for an unsuccessful search: $O(n)$

9. (15%) Show the running time of each algorithm or code segment.
   a. Finding the maximum of an array with $n$ unordered items.
   b. Deleting an item from an $n$-element priority queue that is implemented as a heap.
   c. Print all node values in a binary tree with $n$ nodes by traversing the tree via a preorder scan.
   d. for (i=1; i<n; i++)
      for (j=1; j<i; j++)
          count++;
   e. for (i=1; i<n; i++)
      for (j=1; j<i*i; j++)
          sum+=i+j;

10. (5%) Write the following postfix expression in infix form:
    $5 a * c c * + b d + / + e +$
11. (5%) What is the time complexity of function A? Give sufficient details of your derivation.

```cpp
void A(vector<int>& v, int first, int last)
{
    if (first+1<last){
        int mid=(first+last)/2;
        A(v, first, mid);
        A(v, mid, last);
        B(v, first, mid, last);
    }
    where B(v, first, mid, last) has running time O(n), n=last-first.
}
```

12. (5%) Consider the following insertion sequence:

```
15 57 27 45 30 40 33 39 36 37
```

Draw a red-black tree that corresponds to the data.

13. (5%) Name all statements that are true for a Huffman tree.
   a. A Huffman tree is complete.
   b. Each character is in a leaf node.
   c. Each interior node contains the sum of its children's weights.
   d. Every interior node has exactly two children.
   e. Nodes with a low frequency are near the bottom of the tree.

14. (5%) Show the corresponding adjacency list representation of the undirected graph which has the following adjacency matrix representation.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>7</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
15. (5%) Show the minimum spanning tree of the undirected graph given in Problem 14.

16. (5%) The function `deleteNode()` implements the algorithm that removes the first occurrence of a node in a singly linked list having a specified value. Identify and correct all the errors in this implementation.

```c
void deleteNode(node* front, value target)
{
    node *curr=front, *prev=NULL;
    bool found=false;
    while(curr != NULL && !found)
    {
        if (curr->nodeValue == target)
        {
            prev->next=curr->next;
            delete curr;
            found=true;
        } else {
            curr=curr->next;
            prev=prev->next;
        }
    }
}
```