1. (16%, 4% each) Short questions and multiple choice. For each question you must explain your answer unless it is multiple choice. Write neatly. I cannot give you credit if I cannot read your answer.

   a. What is the advantage of using a binary search tree instead of a binary search algorithm using an array?

      (a) The binary search tree uses less space.
      (b) The binary search tree has $O(\log n)$ for insert, remove and search.
      (c) The binary search tree can grow dynamically while the array is fixed in size.
      (d) The binary search array can only store and search numbers.

   b. What is the best searching algorithm to use if the list of data is not sorted (random).

      (a) First sort the data then do a binary search using an array.
      (b) Linear search
      (c) Binary search using an array.
      (d) Binary search using tree.

   c. Why do we pass a double pointer to the binary tree insert function?

   d. Why do we use a node containing pointer fields to its left child and right sibling for implementing general trees.
2. (16%, 8% each)

a. Find the time complexity in terms of N of the following segment of code. In this case the complexity is the number times the W++ gets executed. This is also the value of W after the code is executed. Note

\[ \sum_{i=1}^{n} k = k(n - m + 1) \text{ and } \sum_{j=1}^{n} i = \frac{n(n + 1)}{2}. \]

for (x = 0; x < N*N; x++)
  if (x % N == 0)
    for (y = x; y > 0; y = y - 5)
      w++;
3. (10%) Show the sequence of comparisons the binary search algorithm goes through to find the number 456 (the algorithm compares 456 to what). Recall that the middle element must be included in one of the two halves. If the middle element falls between 2 elements (even number of elements in the sub list) then choose the element to the right. Use the lazy algorithm (the one that stops only after the list has 1 or 2 items in it).

```
3  5  40  67  92  95  103  150  230  345  456  512  517  656  720  815  932
```

4. (10%) Write a segment of code to remove ALL of the nodes in the linked list. The following structure is used.

```c
struct node {
    int data;
    node *next;
    node *prev;
};
node *front;
```
5. (10%) Trees
   a. Show the following tree after inserting a 7.5, 1.3 and 4.5.

   b. Show the following binary tree after deleting the 9 then the 6.
5. (13%) Write a segment of code to remove ALL of the nodes in the tree.

```c
struct node {
    int data;
    node *Lchild;
    node *Rchild;
}

node *T;
```
6. (15%) Show the following AVL tree after inserting a 7.5.
7. (10%) Having seen the array implementation of linked list, the following is an array implementation of a tree.

a. Given the following array, show the tree it represents.

```c
struct node {
    int Lchild;
    char str[80];
    int Rchild;
}

node Array[8];
```

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>7</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>-1</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

b. Show the array after inserting Joe as the left child of Carlos. Note the empty list still forms a linked list. The Lchild field is ignored and the Rchild field is used as the Next field.