MIDTERM TEST SAMPLE SOLUTION

1. Let $f(n) = 313 \cdot n^9 + 314 \cdot n^8 (\lg n)^2 + 212 \cdot n^3 + 315$. Indicate whether each of the following statements is true or false.

   (a) $f(n) = O(2^n)$  
   True

   (b) $f(n) = \Omega(2^n)$  
   False

   (c) $f(n) = O(n^9)$  
   True

   (d) $f(n) = O(n^8 (\lg n)^2)$  
   False

   (e) $f(n) = \Theta(n^9)$  
   True

   (f) $f(n) = \Theta(n^9 \lg n)$  
   False

   (g) $f(n) = \Omega(n^3)$  
   True

   (h) $n^3 = \Omega(f(n))$  
   False

2. The **Build-Max-Heap** method when run on an array of $n$ elements will convert it into a max-heap.

   (a) What is the time complexity of this algorithm? Give the tightest available bound.  
   $\Theta(n)$ (or $O(n)$)

   (b) Illustrate **Build-Max-Heap** on this array of $n = 10$ items. You may represent the heap as an either an array or a tree. Show intermediate results after each call to **MAX-HEAPIFY**.

   
   
<table>
<thead>
<tr>
<th>5</th>
<th>7</th>
<th>12</th>
<th>8</th>
<th>2</th>
<th>10</th>
<th>6</th>
<th>4</th>
<th>9</th>
<th>11</th>
</tr>
</thead>
</table>

   **MAX-HEAPIFY(5):**

   
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td>12</td>
<td>8</td>
<td>2</td>
<td>10</td>
<td>6</td>
<td>4</td>
<td>9</td>
<td>11</td>
</tr>
</tbody>
</table>

   | 5  | 7  | 12 | 8  | 11 | 10 | 6  | 4  | 9  | 2  |
Figure 1: initial tree

Figure 2: after Max-Heapify(5)

Max-Heapify(4):

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>7</td>
<td>12</td>
<td>8</td>
<td>11</td>
<td>10</td>
<td>6</td>
<td>4</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>7</td>
<td>12</td>
<td>9</td>
<td>11</td>
<td>10</td>
<td>6</td>
<td>4</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 3: after Max-Heapify(4)

Max-Heapify(3): (no change)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>7</td>
<td>12</td>
<td>9</td>
<td>11</td>
<td>10</td>
<td>6</td>
<td>4</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>7</td>
<td>12</td>
<td>9</td>
<td>11</td>
<td>10</td>
<td>6</td>
<td>4</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

Max-Heapify(2):
3. Given a BST $T$ we want to perform a level order of $T$: print the contents of all nodes at depth 0, then all nodes at depth 1, depth 2, etc (each level should be printed from left to right). The root of the tree is $T$.root and a node $p$ has fields $p$.value, $p$.lchild, and $p$.rchild. Give pseudo-code to perform a level-order traversal of $T$, printing the value fields of each node. Use the queue methods enqueue and dequeue.

All we need is a simple breadth-first search using a queue. Almost identical to the reverse level order on p 10 of the week 2 slides.

```python
queue Q
Q.enqueue(T.root)
while (Q not empty)
    p = Q.dequeue
    print p.value
    if (p.lchild is not null)
```
then Q.enqueue(p.lchild)
if (p.rchild is not null)
then Q.enqueue(p.rchild)

4. Consider the following piece of code which, on input $x$ and $n$ (both non-negative integers), returns $x^{2^n}$:

```c
p = x
i = 0
while (i < n)
    p = p*p
    i = i+1
return p
```

You are to prove that this code does indeed return $x^{2^n}$ by the method of loop invariant. An invariant that should work here is $S(p, i)$, which is true iff

$$(i \leq n) \text{ and } (p = x^{2^i})$$

**initialization** Show that prior to entry into the loop, $S(p, i) = S(x, 0)$ is true.

It is true because

- $0 = i \leq n$ ($n$ is given to be a non-negative integer) and
- $x^{2^i} = x^{2^0} = x^1 = x = p$

Therefore $S(x, 0)$ is true.

**maintenance** Show that if $S(p, i)$ is true and the loop is entered, then $S(p', i')$ is true after one more iteration of the loop ($p', i'$ are the new values given to $p, i$ in the loop body).

The loop body does the updates $i' = i + 1$ and $p' = p^2$. From $S(p, i)$ we know that $p = x^{2^i}$ and since we entered the loop we must have $i < n$. Then

- $i < n$ implies $i' = i + 1 \leq n$ and
- $p' = p^2 = (x^{2^i})^2 = x^{2^{i+1}} = x^{2^{i+1}}$

So $S(p', i')$ is also true.

**termination** Show that when exiting the loop, $p$ contains the desired value $x^{2^n}$.

At termination $S(p, i)$ is true and the halting condition $i < n$ is false. $S(p, i)$ says that $i \leq n$, so we must have $i = n$. $S(p, i)$ also implies that $p = x^2$, so at termination we get $p = x^{2^n}$ as claimed.