This quiz has 3 questions, for a total of 10 points.

1. **3 points**  What is a longest common subsequence of TAGC and ATCG? Show your work by writing down the dynamic programming table.

   **Solution:** (2 points for a valid answer. 1 point for a correct table.)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>G</th>
</tr>
</thead>
</table>
   0 | ←0| ←0| ←0| ←0|
   T | ↑0| ↑0| ↓1| ←1| ←1|
   A | ↑0| ↓1| ↑1| ↑1| ↑1|
   G | ↑0| ↑1| ↑1| ↑1| ↓2|
   C | ↑0| ↑1| ↓1| ↓2| ↓2|

   One valid answer is TG:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>G</th>
</tr>
</thead>
</table>
   0 | ←0| ←0| ←0| ←0|
   T | ↑0| ↑0| ↓1| ←1| ←1|
   A | ↑0| ↓1| ↑1| ↑1| ↑1|
   G | ↑0| ↑1| ↑1| ↑1| ↓2|
   C | ↑0| ↑1| ↓1| ↓2| ↓2|

   Another answer is TC:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>G</th>
</tr>
</thead>
</table>
   0 | ←0| ←0| ←0| ←0|
   T | ↑0| ↑0| ↓1| ←1| ←1|
   A | ↑0| ↓1| ↑1| ↑1| ↑1|
   G | ↑0| ↑1| ↑1| ↑1| ↓2|
   C | ↑0| ↑1| ↓1| ↓2| ↓2|

   Yet another is AC:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>G</th>
</tr>
</thead>
</table>
   0 | ←0| ←0| ←0| ←0|
   T | ↑0| ↑0| ↓1| ←1| ←1|
   A | ↑0| ↓1| ↑1| ↑1| ↑1|
   G | ↑0| ↑1| ←1| ↑1| ↓2|
   C | ↑0| ↑1| ↓1| ↓2| ↓2|

2. **4 points**  The following code implements DNA sequence alignment using dynamic programming. The two input sequences are s1 and s2 and F is the dictionary for representing the table of scores. (Don’t worry about traceback.) Fill in the blanks to complete the code.

   ```python
   F[(0, 0)] = ___(a)___
   for i in range(1, m+1):
       F[(i, 0)] = SPACE_PENALTY * i
   for j in range(1, n+1):
       F[(0, j)] = SPACE_PENALTY * j
   for i in range(1, m+1):
       for j in range(1, n+1):
           match = F[___(b)___] + s(___(c)___, s2[j-1])
           delete = F[(i-1, j)] + SPACE_PENALTY
           insert = F[(i, j-1)] + SPACE_PENALTY
           F[(i, j)] = ___(d)___
   ```
Solution: (1 point each.)
(a) 0
(b) (i-1, j-1)
(c) s1[i-1]
(d) max(match, delete, insert)

3. 3 points Given the following array, turn it into a max heap, showing the array and tree representation before and after each Max-Heapify operation.

[1, 2, 3, 4, 5]

Solution: The initial heap as a binary tree: (1 point)

```
      1
     / \
    2   3
   / \ / \  
  4   5 5 4
```

After applying Max-Heapify to 2 at index 1: (1 point)

```
      1
     / \  
    5   3
   / \ / \  
  4   2 1 2
```

After applying Max-Heapify to 1 at index 0: (1 point)

```
      5
     /  
    4   3
   / \ /  
  1   2 1 2
```