(a) Show by induction:

- Let \( l \) = the length of array, \( m = |l-3| \)
- Show that an array of length \( l \) is sorted after running our algorithm
- For \( m = 1 \):
  - \( l = 1 \), the element is sorted
  - \( l = 2 \), after swapping, this part is sorted
  - \( l = 3 \), this is similar to an insertion sort on array length of 3 (why?)
Question 1

- Assume $m=n$, this algorithm is correct.
- For $m=n+1$:
  - $m \% 3 = 1$, we will put the smallest element on the first place and ...
  - $m \% 3 = 2$, just the same as $m \% 3 = 1$
  - $m \% 3 = 0$, this is a little complicated...
We have 3 routines sorting 2/3 total elements, by assumption, these routines correctly sort these elements.
Question 1

(b) \[ T(n) = 3T(2n/3) + O(n) \]

(c) Master Theorem can easily give a tight bound
Question 2

- An ordinary RadixSort can handle this.
Question 3

"Bucketing by Length":

Put each item in the bucket that represents its length.
Use RadixSort to sort each bucket.

Bucket 3 stores all length-3 items.
Take out items one by one from 1st bucket, 2nd bucket, ...
Question 3

- Why is this algorithm correct?
- How to analyze the time complexity?
Question 4

(a) Easy, free points

(b) By the characteristic of Young Tableau, if Y[1,1] is infinity then no other elements can be greater than it, so ...
Question 4

(c) Use a Heap-like method, maximum # of swapping steps = m+n-1

- To show correctness, use induction to show that, after each swap:
  (I) at most one entry may violate Young Tableau property
  (II) If an entry $X$ violates the property, the entries above or left of $X$ have values smaller than the entries down or right of $X$
Question 4

(d)

- First insert each unsorted elements into the tableau
  - $O(n^2(n+n)) = O(n^3)$
- Then use Extract-Min $n^2$ times
  - $O(n^2(n+n)) = O(n^3)$
- Total:
  - $O(n^3) + O(n^3) = O(n^3)$
Question 5 (Bonus)

Let $\text{Bound}_i(K)$ = position of the boundary on $i^{th}$ row which pivots $K$

\[
\begin{array}{cccc}
1 & 2 & 4 & 6 \\
3 & 5 & 7 & 8 \\
6 & 8 & 9 & 12 \\
9 & 10 & 11 & 13 \\
\end{array}
\]

$\text{Bound}_1(5)=3$
Question 5

- Observe that for any K,
  \[ \text{Bound}_i(K) \geq \text{Bound}_{i+1}(K) \quad \text{(Why?)} \]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td>5</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>8</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>10</td>
<td>11</td>
<td>13</td>
</tr>
</tbody>
</table>
Question 5

- Then we search for the element just before the boundary

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>9</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>11</td>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>
Question 5

- How about its complexity?
- Suppose there’re $n$ rows, $m$ columns
- Finding $\text{Bound}_1(K)$ - $O(n)$
- Search for $\text{Bound}_i(K)$ from tail - $O(n+m)$
- Find $K$ - $O(m)$
- Total: $O(n) + O(n) + O(m) = O(n+m)$