1. Run Dijkstra's shortest path algorithm in the following graph with vertex S as the source, and fill the table below with the results.

Name & UW NetID:

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Distance</th>
<th>Predecessor</th>
<th>Processed</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>0</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>S</td>
<td>✓</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>S</td>
<td>✓</td>
</tr>
<tr>
<td>T</td>
<td>6</td>
<td>B, E</td>
<td>✓</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>B</td>
<td>✓</td>
</tr>
</tbody>
</table>
2. Following is the pseudocode for Dijkstra’s shortest path algorithm with binary heap. Do runtime analysis of this code. (Annotate the pseudocode with big-$O$ values next to relevant statements/loops in the code.)

```plaintext
1: function Dijkstra(Graph G, Vertex source)  
2:     initialize distances to $\infty$ \textcolor{red}{$\leftarrow O(|V|)$} 
3:     source.dist = 0 \textcolor{red}{$\leftarrow O(1)$} 
4:     mark all vertices unprocessed \textcolor{red}{$\leftarrow O(|V|)$} 
5:     initialize MPQ as a min priority queue 
6:     add source with priority 0 
7:     while MPQ is not empty do \textcolor{red}{$\leftarrow O(|V|)$} \textcolor{red}{$\leftarrow O(|V|)$} 
8:         u = MPQ.getMin() \textcolor{red}{$\leftarrow O(|V|)$} 
9:         for each edge (u,v) leaving u do \textcolor{red}{$\leftarrow O(|E|/|V|)$} 
10:            if u.dist + w(u,v) < v.dist then 
11:                if v.dist == $\infty$ then \textcolor{red}{$\leftarrow O(|E|)$} 
12:                    MPQ.insert(v, u.dist + w(u, v)) \textcolor{red}{$\leftarrow O(|V| \log |V|)$} 
13:                else \textcolor{red}{$\leftarrow O(|V| \log |V|)$} 
14:                    MPQ.decreasePriority(v, u.dist + w(u,v)) 
15:                end if \textcolor{red}{$\leftarrow O(|V| \log |V|)$} 
16:                v.dist = u.dist + w(u,v) \textcolor{red}{$\leftarrow O(1)$} 
17:                v.predecessor = u \textcolor{red}{$\leftarrow O(1)$} 
18:            end if \textcolor{red}{$\leftarrow O(|E|)$} 
19:         end for \textcolor{red}{$\leftarrow O(|E|)$} 
20:     mark u as processed \textcolor{red}{$\leftarrow O(1)$} 
21:     end while \textcolor{red}{$\leftarrow O(|V|)$} 
22: end function
```

3. Circle the strong connected components in the following graph.

```
A -- B -- D -- E
|    |    |    |
C -- F -- J
|
K
```

Total time complexity: $O(|V| \log |V| + |E| \log |V|)$