Q1  
a)  
Priority Queue stores a collection of entries, (key and value).  
The aim of priority queue, is to give each entry a key value, relating to how important that entry is.  
So if something is accessed many times, it has a important (low) key value, allowing faster access times.  
  
min: O(1)   
removeMin: O(1)  
put: O(n)  
**Arbitory inserts  
4/5**b)  
Correct 15/15

**Remember that heaps don’t have smaller/bigger for left/right – they only have smallest is parent.**

**Q2**

a)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 2 | 16 | 30 |  |  |  |  |  | 74 | 9 |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 2 | 16 | 30 | 15 |  |  |  |  | 74 | 9 |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 2 | 16 | 30 | 15 | 15 |  |  |  | 74 | 9 |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 2 | 16 | 30 | 15 | 15 | 45 |  |  | 74 | 9 |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|  |  | 2 | 16 | 30 | 15 | 15 | 45 |  |  | 74 | 9 | 23 |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 2 | 16 | 30 | 15 | 15 | 45 |  | 22 | 74 | 9 | 23 |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 52 |  | 2 | 16 | 30 | 15 | 15 | 45 |  | 22 | 74 | 9 | 23 |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 52 |  | 2 | 16 | 30 | 15 | 15 | 45 | 6 | **74** | **9** | **23** | **22** |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

**Make sure put correct num in correct blocks from test.**

**b)**

No, **REASON: Small array, unnecessary wrap arounds (1/2)**

**c)**

14

Q3  
a)  
public V remove(K k)

{

for(int i = 0; i < size; i++)

{

K tempK;

V tempV;

if(A[i].getKey().equals(k)

{

tempV = A[i].getValue();

for(int j = i; j < size-1; j++)

{

A[j] = A[j+1];

}

A[size-1].setKey(null);

A[size-1].setValue(null);

A[size-1] = null;

size--;

return tempV;

}

}

return null;

}

**Removes a entry, not key. Else good**

b)

**POLYNOMIAL ACCUMULTION – look at notes**

**c)**

**A picture containing diagram, technical drawing, plan, line

Description automatically generated  
A picture containing diagram, technical drawing, plan, schematic

Description automatically generated**

**Q**4  
a) 45  
b)Add 39 to Right of 38  
c) Add 8 to Left of 8  
d) MOVE 38 to Right of 33  
e)Move 8 To Left of 33 **X do 14  
f) O(n)**  
g)O(n)  
h)Yes