**Rubric**

| Ensure code works if list is empty | 1 |
| --- | --- |
| Initialize a head and tail for each list containing the numbers less than x and greater than or equal to x | 2 |
| Traverse the list | 3 |
| Nodes follow relative order | 4 |
| Ensure code works if list contains only numbers greater than x or less than it | 2 |
| Connect both lists such that all numbers less than x are at the beginning | 2 |
| Return the head of the modified list at the end | 1 |

**Set A**

**Python solution**

def partition(head, x):

if(head == None):

return None

lesser = None

higher = None

highStart = None

lessStart = None

while(head != None):

if(head.val < x):

if(lesser == None):

lesser = head

lessStart = head

else:

lesser.next = head

lesser = lesser.next

else:

if(higher == None):

higher = head

highStart = head

else:

higher.next = head

higher = higher.next

head = head.next

if(lesser == None):

return highStart

elif(higher == None):

return lessStart

higher.next = None

lesser.next = highStart

newHead = lessStart

return newHead

**Java solution**

public Node rearrangeNodes(Node head, int x) {

if(head == null) {

return null;

}

Node lesser = null;

Node higher = null;

Node highStart = null;

Node lessStart = null;

while(head != null) {

if((int) head.elem < x) {

if(lesser == null) {

lesser = head;

lessStart = head;

} else {

lesser.next = head;

lesser = lesser.next;

}

} else {

if(higher == null) {

higher = head;

highStart = head;

} else {

higher.next = head;

higher = higher.next;

}

}

head = head.next;

}

if(lesser == null) {

return highStart;

} else if(higher == null) {

return lessStart;

}

higher.next = null;

lesser.next = highStart;

Node newHead = lessStart;

return newHead;

}

**Set B**

**Python solution**

def partition(head, x):

if(head == None):

return None

lesser = None

higher = None

highStart = None

lessStart = None

while(head != None):

if(head.val < x):

if(lesser == None):

lesser = head

lessStart = head

else:

lesser.next = head

lesser = lesser.next

else:

if(higher == None):

higher = head

highStart = head

else:

higher.next = head

higher = higher.next

head = head.next

if(lesser == None):

return highStart

elif(higher == None):

return lessStart

higher.next = None

lesser.next = highStart

newHead = lessStart

return newHead

**Java solution**

public Node rearrangeNodes(Node head, int x) {

if(head == null) {

return null;

}

Node lesser = null;

Node higher = null;

Node highStart = null;

Node lessStart = null;

while(head != null) {

if((int) head.elem > x) {

if(lesser == null) {

lesser = head;

lessStart = head;

} else {

lesser.next = head;

lesser = lesser.next;

}

} else {

if(higher == null) {

higher = head;

highStart = head;

} else {

higher.next = head;

higher = higher.next;

}

}

head = head.next;

}

if(lesser == null) {

return highStart;

} else if(higher == null) {

return lessStart;

}

higher.next = null;

lesser.next = highStart;

Node newHead = lessStart;

return newHead;

}