(1) Write a function called partition which accepts an integer array A, its left and right bounds (indices), and a number p. The function 'partitions' the array A around the number p. For example, if A = {3, 1, 0, 7, 2, 10, -1, 9, 6, 2}, and left = 0 and right=9, and let's assume p = 4. Then after partition, A = {3, 1, 0, 2, -1, 2, 7, 10, 9, 6}. All numbers below or equal p are to the of all numbers above p. The function should return the index of the last number in the left partition. For instance in this case it should return: 5 (the index of the second 2). int partition(int A[], int left, int right){ (2) Describe in one line what the following code does. void recursive(int A[], int left, int right){ What does it do?	Roll No	S	ec
(1) Write a function called partition which accepts an integer array A, its left and right bounds (indices), and a number p. The function 'partitions' the array A around the number p. For example, if A = {3, 1, 0, 7, 2, 10, -1, 9, 6, 2}, and left = 0 and right=9, and left's assume p = 4. Then after partition, A = {3, 1, 0, 2, -1, 2, 7, 10, 9, 6}. All numbers below or equal p are to the of all numbers above p. The function should return the index of the last number in the left partition. For instance in this case it should return: 5 (the index of the second 2). int partition(int A[], int left, int right){ (2) Describe in one line what the following code does. void recursive(int A[], int left, int right){ What does it do?			
(1) Write a function called partition which accepts an integer array A, its left and right bounds (indices), and a number p. The function 'partitions' the array A around the number p. For example, if A = {3, 1, 0, 7, 2, 10, -1, 9, 6, 2}, and left = 0 and right=9, and let's assume p = 4. Then after partition, A = {3, 1, 0, 2, -1, 2, 7, 10, 9, 6}. All numbers below or equal p are to the of all numbers above p. The function should return the index of the last number in the left partition. For instance in this case it should return: 5 (the index of the second 2). Integration integration in the left, int right) { (2) Describe in one line what the following code does. void recursive(int A[], int left, int right) { What does it do? if(left <right) td="" {<=""><td></td><td>Qui</td><td>z 3</td></right)>		Qui	z 3
(indices), and a number p. The function 'partitions' the array A around the number p. For example, if A = {3, 1, 0, 7, 2, 10, -1, 9, 6, 2}, and left = 0 and right=9, and let's assume p = 4. Then after partition, A = {3, 1, 0, 2, -1, 2, 7, 10, 9, 6}. All numbers below or equal p are to the of all numbers above p. The function should return the index of the last number in the left partition. For instance in this case it should return: 5 (the index of the second 2). int partition(int A[], int left, int right){ (2) Describe in one line what the following code does. void recursive(int A[], int left, int right){ What does it do?			10 + 5 points.
(2) Describe in one line what the following code does. void recursive(int A[], int left, int right){ if(left <right){ do?<="" does="" it="" td="" what=""><td>(indices), and a number example, if A = {3, 1, 0}. Then after partition, A = of all numbers above p</td><td>r p. The function 'partition', 7, 2, 10, -1, 9, 6, 2}, an = {3, 1, 0, 2, -1, 2, 7, 10, . The function should re</td><td>ons' the array A around the number p. For and left = 0 and right=9, and let's assume p = 4. 9, 6}. All numbers below or equal p are to the left turn the index of the last number in the left</td></right){>	(indices), and a number example, if A = {3, 1, 0}. Then after partition, A = of all numbers above p	r p. The function 'partition', 7, 2, 10, -1, 9, 6, 2}, an = {3, 1, 0, 2, -1, 2, 7, 10, . The function should re	ons' the array A around the number p. For and left = 0 and right=9, and let's assume p = 4. 9, 6}. All numbers below or equal p are to the left turn the index of the last number in the left
void recursive(int A[], int left, int right){ if(left <right){ do?<="" does="" it="" td="" what=""><td>int partition(int A[], int le</td><td>eft, int right){</td><td></td></right){>	int partition(int A[], int le	eft, int right){	
void recursive(int A[], int left, int right){ if(left <right){ do?<="" does="" it="" td="" what=""><td></td><td></td><td></td></right){>			
void recursive(int A[], int left, int right){ if(left <right){ do?<="" does="" it="" td="" what=""><td></td><td></td><td></td></right){>			
void recursive(int A[], int left, int right){ if(left <right){ do?<="" does="" it="" td="" what=""><td></td><td></td><td></td></right){>			
void recursive(int A[], int left, int right){ if(left <right){ do?<="" does="" it="" td="" what=""><td></td><td></td><td></td></right){>			
void recursive(int A[], int left, int right){ if(left <right){ do?<="" does="" it="" td="" what=""><td></td><td></td><td></td></right){>			
void recursive(int A[], int left, int right){ if(left <right){ do?<="" does="" it="" td="" what=""><td></td><td></td><td></td></right){>			
void recursive(int A[], int left, int right){ if(left <right){ do?<="" does="" it="" td="" what=""><td>(2) Describe in one line</td><td>what the following and</td><td>a does</td></right){>	(2) Describe in one line	what the following and	a does
if(left <right){< td=""><td>. ,</td><td></td><td></td></right){<>	. ,		
int p = A[right];	if(left <right){ int p = A[right];</right){ 		What does it do?
<pre>int q = partition(A, left, right); recursive(A, left, q-1); recursive(A, q+1, right); };</pre>	recursive(A, left, q	-1);	