```
public static void Problem4A(int[] a) {

	// pick the largest value of the current 3 indices and swap around them
	// O(N)
	for(int i = 1; i < a.length; i = i+2) {
		int x = i-1;
		int y;
		if( i != a.length-1)
		y = i+1;
		else
			y = i;
		int z = Math.max(a[x],Math.max(a[i],a[y]));
		int index = 0;

		if(z == a[x]) index = x;
		else if (z == a[y]) index = y;
		else index = i;

		if(index != i) {
			int temp = a[i];
			a[i] = a[index];
			a[index] = temp;
		}

	}

}
```

1. This code runs a single loop that checks chunks of 3 – i-1,I, and i+1 and increments I by 2. It searches for whichever the maximum value is and swaps it to the midpoint (which would be 'I'). The loop will run in $O(N)$ time since it's a single loop bounded by the length of the input array.
2. This solution is a linear time one. However, a different solution would be to sort the array first and swap the elements ahead by 1 to get the scrambled array. This would be $NlogN + N$ time.