Quick Sort

// Pre-condition: low and high are value indices into numbers.
// Post-condition: The values in numbers will be sorted in between indices low and high
void quicksort(int* numbers, int low, int high) {
    // Only have to sort if we are sorting more than one number
    if (low < high) {
        int split = partition(numbers, low, high);
        quicksort(______________________________);
        quicksort(______________________________);
    }
}

// Swaps the values pointed to by a and b.
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

vals

8 3 6 9 2 4 1 0 7 5

If we call quicksort(vals, 0, 9) (assume 6 is the partition element) fill in split and what the following recursive calls would contain:

split = _______________________
quicksort(__________________________)
quicksort(__________________________)

Assume the 1st time partition is called, i = 2. Show the contents of vals after each iteration of the while loop:

After 1st Loop:

After 2nd Loop:

After 3rd Loop:

After putting partition in the right spot:

// Returns the partition index such that all the values stored in vals from low // to partition are < partition & all the vals from partition to high are > .

```c
int partition(int* vals, int low, int high) {
    int temp;
    int i, lowpos;

    if (low == high) return low; // A base case that should never really occur.

    // Pick a random partition element and swap it into index low.
    i = low + rand()%(high-low+1);
    temp = vals[i];
    vals[i] = vals[low];
    vals[low] = temp;

    lowpos = low; // Store the index of the partition element.
    low++;

    while (low <= high) {
        // Move the low pointer until we find a value too large for this side.
        while (____________________________________) low++;

        // Move high until we find a value too small for this side.
        while (____________________________________) high--;

        if (low < high) // Swap the two values that were on the wrong side.
            swap(&vals[low], &vals[high]);
    }

    swap(&vals[lowpos], &vals[high]); // Swap partition into right spot.

    return high; // Return the index of the partition element.
}
```