Problem 1.

Implement quicksort for sorting arrays of integer numbers. The input can be generated by a loop:

```c
int i, j, m, n;
n = 10*1000*1000;
int *S = (int*) malloc(n*sizeof(int));
srandom(2009);
for (i = 0; i < n; i++)
    S[i] = random();
```

After sorting compute, for every integer \( k \), how many times you have \( S[i+1] - S[i] = k \), and the output should inform about that in the following format:

```plaintext
diff 0 count 3246
diff 1 count 3352
diff 2 count 3206
```

Print only the positive counts.

Before the submission, you should test your program. You should try the simplest possible implementation, and at least two of the following possible improvements:

a. Use median of five for pivot selection if the array fragment has at least 50 elements.
b. Use median of three for pivot selection if the array fragment has at least 50 elements.
c. Use insertion sort for if the array fragment is smaller than 10 keys.
d. For array fragments smaller than 50 elements, use mergesort.
e. For array fragments smaller than 500 elements, use mergesort.
f. Eliminate the recursive call with the larger of the two fragments that occur in recursive calls.

Problem 2.

Implement and test mergesort with the same loop for generating the input and the same output.

You should email the source code of two programs, and it will be compiled and run in a Unix/Linux system. Attach also a short report describing in one sentence each program you have tested (test 3 versions of quicksort and one of mergesort, but submit only the fastest of your quicksort programs) and a little table of the run time of the programs you have tested. of the running times.

I will post more instructions once I test the homework myself. Start with testing on small inputs, say, \( n = 10*1000 \).