Question 1: Show me the result of partitioning $\{251, 983, 036, 783, 734, 186, 494, 469, 619\}$ using the "median of three" pivot selection. Give you answer as numbers each separated by a single space.

Question 2: Below is a dynamic program **gc()**, which takes a string of integer digits, like "12345", and a boolean, **zero**. It returns a long long. It is correct, but it only implements "Step 1" of dynamic programming. Your job is to do "Step 2" -- memoize the program.

In the text box of the answer, simply list what changes you'd make, such as:

```
Before line 9, add "int j".
```

List one change per line. I want to stress that you don't need to figure out what this code is doing. You do need to understand dynamic programming to the point where you can memoize this code correctly, even if you don't really know what it does.

```
/* Line  1 */ class XXX {
/* Line  2 */   public:
/* Line  3 */     long long gc(string s, bool zero);
/* Line  4 */ };
/* Line  5 */
/* Line  6 */ long long XXX::gc(string s, bool zero)
/* Line  7 */ {
/* Line  8 */   int i, f, nz;
/* Line  9 */   string r, news;
/* Line 10 */   long long rv, one, rplus1;
/* Line 11 */
/* Line 12 */   if (s.size() == 0) return 0;
/* Line 13 */
/* Line 14 */   one = 1;
/* Line 15 */   rv = 0;
/* Line 16 */   if (s.size() == 1) {
/* Line 17 */     for (i = 1; i <= s[0]-'0'; i++) rv ^= (1L << i);
/* Line 18 */     if (zero) rv ^= 1;
/* Line 19 */     return rv;
/* Line 20 */   }
/* Line 21 */
/* Line 22 */   f = s[0] - '0';
/* Line 23 */   for (i = 1; i < s.size() && s[i] == '0'; i++) ;
/* Line 24 */   nz = i-1;
/* Line 25 */   r = s.substr(nz+1);
/* Line 26 */   rplus1 = (r == "") ? 0 : atoi(r.c_str());
/* Line 27 */   rplus1++;
/* Line 28 */   rv ^= ((1L << f)*rplus1);
/* Line 29 */   rv ^= rplus1;
/* Line 30 */
/* Line 31 */   if (r != "") rv ^= gc(r, false);
/* Line 32 */
/* Line 33 */   if (s[0] == '0') return rv;
/* Line 34 */
/* Line 35 */   if (zero || s[0] > '1') news.push_back(s[0]-1);
/* Line 36 */   for (i = 1; i < s.size(); i++) news.push_back('9');
/* Line 37 */   return rv ^ gc(news, zero);
/* Line 38 */ }
```

Question 1:

```
The vector starts as: 251, 983, 036, 783, 734, 186, 494, 469, 619

The median of 251, 734 and 619 is 619.  Swap it with 251:

619, 983, 036, 783, 734, 186, 494, 469, 251

Now do the partition:

619, 983, 036, 783, 734, 186, 494, 469, 251
     ^^^                                ^^^
619, 251, 036, 783, 734, 186, 494, 469, 983
               ^^^                 ^^^
619, 251, 036, 469, 734, 186, 494, 783, 983
                    ^^^       ^^^
619, 251, 036, 469, 494, 186, 734, 783, 983
                              ^^^
Swap element zero and the 186 and you get your answer:

186, 251, 036, 469, 494, 619, 734, 783, 983
```

Question 2: First you need to add a cache. I would add an unordered_map whose keys are strings and whose vals are long longs. So the first part of my answer would be:

```
After line 3, add 'unordered_map <string, long long> cache'
```

Next, I need to make a memoization key from **s** and **zero**. Since **s** is only composed of digits, you can simply turn **zero** into a cahracter and append it to **s**. This would be my code for that:

```
After line 10, add 'string key'
After line 12, add 'key = s + ((zero) ? "A" : "B")'
```

Now, you need to check the cache. This can be before line 12, after line 12, or after line 20. I would probably do it after line 20, so all the base cases run before I mess with the cache (I'd also set the key after line 20, rather than after line 12 as above. Either is ok):

```
After line 20, add "if (cache.find(key) != cache.end() return cache[key];"
```

There are two places that you return, and in both places, you need to set the cache, so:

```
In the body of the if statement in line 33, before the return, add cache[key] = rv;
On line 47, set rv = rv ^ gc(news, zero), and then do cache[key] = rv; return rv;
```