```
1.      typedef struct Q {
2.        int* q;
3.        int pos;
4.        int size;
5.      } Q;
6.
7.      struct Q* initQ(int n) {
8.        int i;
9.        struct Q *newQ = (struct Q *) malloc(sizeof(Q));
10.       newQ->q = (int*) malloc(sizeof(int)*n);
11.       newQ->pos = -1;
12.       newQ->size = n-1;
13.     }
14.
15.     void putWork(struct Q* workQ) {
16.       if (workQ->pos < (workQ->size)) {
17.         workQ->pos++;
18.         workQ->q[workQ->pos] = (int) (rand( )%2*(workQ->pos/(workQ->size/10));
19.       } else printf("ERROR: attempt to add Q element%d\n", workQ->pos+1);
20.     }
21.
22.     int getWork(struct Q* workQ) {
23.       if (workQ->pos > -1) {
24.         int w = workQ->q[workQ->pos];
25.         workQ->pos--;
26.         return w;
27.       } else printf("ERROR: attempt to get work from empty Q%d\n", workQ->pos);
28.     }
```

I'll first step through the syntax on line 18, which is one of the most complicated statements. *workQ* is a pointer to a *Q* struct. *workQ->q* points to the array of integers that are the elements of the queue. *workQ->pos* points to the *pos* member of the *Q* struct pointed to by *workQ*. All of these together access the pos element of the *q* that is part of the *Q* struct pointed to by *workQ*.

Lines 1 - 5 define a new type that is a structure containing the declared fields. This represents a work queue, where *size* is the max number of elements, *pos* is the current last added element position in the queue, and *q* is a pointer to the list of integers that make up the elements in the queue.

Line 9 allocates on the heap an instance or copy of the structure. *newQ* is a pointer to the allocated structure. Lines 10 - 12 initialize the three fields of the structure.

The function at line 15 puts a piece of work onto the queue. Line 16 checks the *pos* field of the queue. If it's value is less than the value of the size field than another

element can be added to the queue, otherwise there is an error because the list of integers pointed to by *q* is full. Line 17 increments the position (*pos*) to add a element from the position of the last added element to an empty element in the array pointed to by *q*. Line 18 sets the value of the element to a random integers. Note that the bold text is different than what is in hw3Q.c and basically distributes work in the same way that hw3Loop.c does.