University of Massachusetts Dartmouth
Department of Electrical and Computer Engineering

ECE 161 Submit Name: `queue100.cpp` (or `queue150.cpp` or `queue200.cpp`)
Project 7 Due: see http://ece161.viall.org

The Dynamic Stack program you have recently written allows users to put (via push), or get (via pop) information onto a stack. The stack is a “Last In First Out” (LIFO) type of data structure. That is, the most recently pushed item will be the first item popped.

As you know, a queue is similar to a stack, except a queue is a “First In First Out” (FIFO) type of data structure. The operation “enqueue” is analogous to “push”, and the operation “dequeue” is analogous to “pop”. Using the attached sample output and displayed structures as a guide, implement the functions CreateQ (), DeleteQ (), EnQ (), DeQ (), and DisplayQ. CreateQ()/DeleteQ must allocate/free memory for INFO. EnQ/DeQ must allocate/free memory for each QNODE. In this handout, labels "front" and "rear" are used instead of "head" and "tail" as in the previous program. Both are used...the KEY is to know what each one specifically does. When the program exits, your code must free all dynamically allocated memory, even if there are still items in the queue.

Some type of a command structure must be implemented in the main() function, where 0=Exit, 1=Enqueue, 2=Dequeue, 3=Display Queue.

For 50 extra points on the project (max 150), implement a command interface which used alphabetic commands instead of numbers. Upper and lower case commands are considered equivalent. The command interface must successfully parse any partially entered command, so long as it is a legal command. If you impliment this feature, name the submitted file queue150.cpp

For an additional 50 points (max 200): In the case of the "enqueue" command, if the user enters the item on the same line as the "enqueue" command, it must parse. If the user types "enqueue" without any item after it, the program must prompt for the item. If you implement this feature, name the submitted file queue200.cpp. The strtok() function provides some useful functionality

Sample run (`queue100.cpp`):

```
Welcome to Queue Program
Commands:  0 - exit; 1 - enqueue; 2 - dequeue; 3 - display
Command: 2
Dequeue= ?Error - Queue empty
  Queue= Empty
Command: 1
   Item: 53
  Queue= 53
Command: 1
   Item: 32
  Queue= 53 32
Command: 6
?Invalid command
Command: 1
   Item: 80
  Queue= 53 32 80
Command: 2
Dequeued=53
  Queue= 32 80
Command:0
```

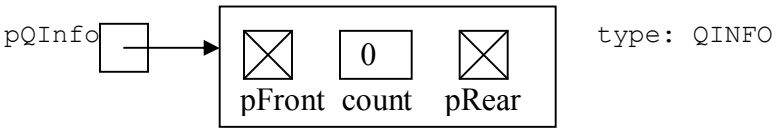
Sample run (queue150.cpp)

```
Welcome to the Queue Program
Commands:  exit, enqueue,
           dequeue, display
Command: dequ
Dequeue= ?Error - Queue empty
  Queue= Empty
Command: enqueue
   Item: 53
  Queue= 53
Command: enQUEue
   Item: 32
  Queue= 53 32
Command: en
   Item: 80
  Queue= 53 32 80
Command: dis
  Queue= 53 32 80
Command: disply
?Invalid command
Command: display
  Queue= 53 32 80
Command: dequeue
Dequeue= 53
  Queue= 32 80
Command: e
?Invalid Command
Command: ex
```

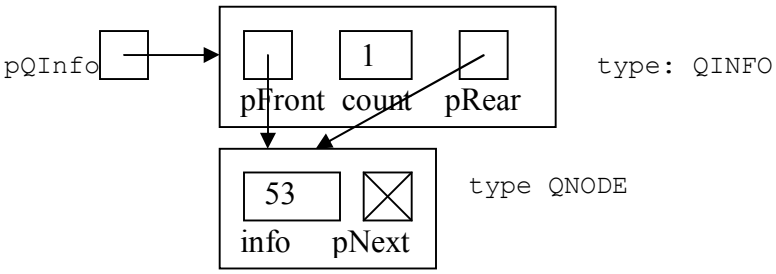
Sample run (queue200.cpp)

```
Welcome to the Queue Program
Commands:  exit, enqueue,
           dequeue, display
Command: dequ
Dequeue= ?Error - Queue empty
  Queue= Empty
Command: enqueue
   Item: 53
  Queue= 53
Command: enQUEue 32
  Queue= 53 32
Command: en 80
  Queue= 53 32 80
Command: dis
  Queue= 53 32 80
Command: disply
?Invalid command
Command: display 13
%Extra characters ignored
  Queue= 53 32 80
Command: dequeue
Dequeue= 53
  Queue= 32 80
Command: e
?Invalid Command
Command: ex
```

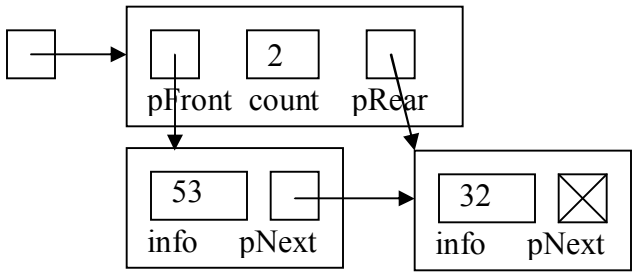
```
void main()
{
      QINFO *pQInfo;
      int DataIn, DataOut, Code

      pQInfo = CreateQ();
```



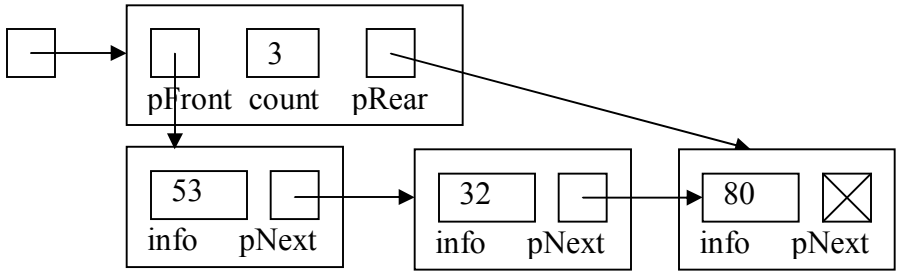

```
      DataIn=53;
      Code = EnQ(pQInfo, DataIn);
```




```
      DataIn=32;
      Code = EnQ(pQInfo, DataIn);
```




```
      DataIn=80;
      Code = EnQ(pQInfo, DataIn);
```




```
      Code = DeQ(pQInfo, &DataOut);
```