Name: ______________________________ Student #: __________________

Engineering 4892 — Data Structures

Quiz 1

Dr. D. K. Peters

June 5, 2002

Instructions: Answer all questions. Write your answers on this paper. This is a closed book test, no textbooks, notes, calculators or other aides are permitted.
Total points: 50

---

1. [10 points] Give a brief definition for each of the following as it is used in this course.

   a) Pre-condition

   b) Post-condition

   c) Abstract data type

   d) Interface

   e) Module

2. [20 points] Below is the declaration of a Queue class template. Implement the functions, as specified on the following pages.

```
template <class T>
class Queue
{
  public:
  // A queue (FIFO list) of type T.
  // Modeled by:
  //   Q sequence of T

  // Local types
    enum Status { Ok, NewFail };

  // Constructors
    Queue();                  // Post: Q = _

    Queue(const Queue<T>& r); // Post: Q' = r.Q
  // Destructor
    ~Queue();

  // Accessors
    T front() const;
    // Pre: |Q| > 0
    // Post: Result = first element of Q

    bool empty() const;       // Post: Result = (|Q| = 0)

    int size() const;         // Post: Result = |Q|

    Status getStatus() const { return err; }
    // Post: Result = status of last access

  // Mutators
    Status enque(T x);
    // Post: Result = Ok -> Q’ = Qx

    void deque();
    // Pre: |Q| > 0
    // Post: Q’ = Q with first element removed

    Queue<T>& operator=(const Queue<T>& r);
    // Assignment operator
    // Post: this is a copy of r

  private:
    // Representation:
    // head != 0 -> Q = { head->data, head->next->data, ...  } /\
    // head = 0 -> Q = _
    struct Node {
      T data;
      Node* next;
    };
    Node* head;   // Points to the head of the queue.
    Status err;   // Status of the last call.

    // Private methods
    void copyList(const Node* src, Node** dest);
    void deleteList(Node* cur);
};
```

a) [10 points]

```
/****************************************************************
 * size -- return the length of the Queue
 *
 * Post: Result = |Q|
 ****************************************************************/
template <class T>
int
Queue<T>::size() const
{
```

b) [10 points]

```
/****************************************************************
 * deque -- remove the element from the front of the Queue
 *
 * Pre: head != 0
 * Post: head' = head->next
 ****************************************************************/
template <class T>
void
Queue<T>::deque()
{
```

3. [20 points] A common problem in many kinds of text processing (e.g., compilers) is to check that brackets are in matching, properly nested, pairs (e.g., "`({}[]([]))`" is properly nested, but "`({)}`" is not). Give an implementation of a function, as specified below, that uses one or more (STL) `Stack`s or `List`s to check all of the brackets found in the given string and return `true` if they are in matching and properly nested pairs, and `false` if they are not.

```
/****************************************************************
 * matchBrackets -- determine if the brackets are matched in exp.
 *
 * Pre: exp contains an expression to be checked, possibly including
 *      rounded (), square [] or curly {} brackets (any other characters in
 *      exp can be ignored).
 * Post: Result = true if all of the brackets in exp are in matched and
 *                properly nested pairs.
 *       Result = false otherwise.
 ****************************************************************/
bool
matchBrackets(const string& exp)
{
```