December 10, 2009 Name ______________________________

EECS 311 Data Structures Final Exam

Don't Panic!

1. [10 pts] Show how Quicksort would sort the array below. Pick the pivot with median-of-three, using integer division to get the center. Don't sort the three, just swap the pivot with the last element. Be very clear about what goes where in each partitioning phase, e.g., write something like the following for each partitioning:

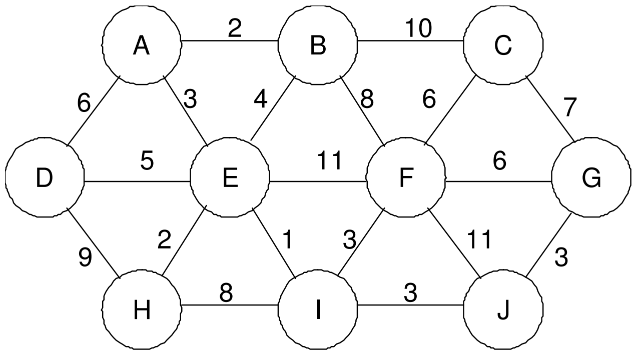
sorting from __ to __, pivot __ swaps with __
quicksort pass swaps __ with __, __ with __, __ with __, ...
result = ...

Circle the pivot in its final location. When a partition is 3 elements or fewer, just indicate the swaps needed, if any, to directly sort it.

0	1	2	3	4	5	6	7	8	9	10
23	**51**	**4**	**18**	**8**	**72**	**31**	**42**	**17**	**9**	**5**

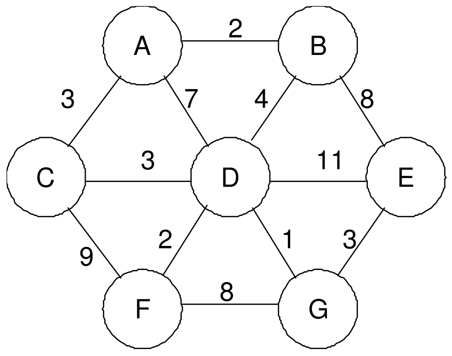
December 10, 2009 Name ______________________________

2. [10 pts] In the table cells below, show the values that **Prim's** algorithm would find while creating a minimum spanning tree for the graph below starting from vertex D. Initial values are shown. Under **best edge weight** and **best edge vertex** put the sequence of weights and vertices of the best edge leading to the given vertex, in the order found. Under **when done** put 1 for the 1st edge finished, 2 for the 2nd, and so on. Draw a line on the graph edges used in the **final** MST. Is this unique? Why or why not?



vertex	when done	best edge weight	best edge vertex
A		∞	-
B		∞	-
C		∞	-
D	0	0	-
E		∞	-
F		∞	-
G		∞	-
H		∞	-
I		∞	-
J		∞	-

3. [10 pts] In the table below show the values Dijkstra's algorithm would generate to find the shortest path from A to G. Under **best path** put the sequence of path distances found, in order, for each vertex. Under **best vertex** put the path vertices found, in order. Under **when done** put 1 for the first vertex that is finished, 2 for the 2nd vertex finished, etc.



vertex	when known	best path	best vertex
A	0	0	-
B		∞	
C		∞	
D		∞	
E		∞	
F		∞	
G		∞	

4. [10 pts] Use the dynamic programming approach to sequence alignment for the problem below. Matching letters score +4 points, mismatching letters score -3, and a gap in either sequence scores -2. Draw small arrows from each cell X to the previous cell(s) whose score leads to the one in X. Show an optimal alignment that follows from this table. Is it unique? Why or why not?

Sequence 1: G T A T C G A
Sequence 2: G A T C G A A

		G	**T**	**A**	**T**	**C**	**G**	**A**
G								
A								
T								
C								
G								
A								
A								

5. [10 pts] The dynamic programming formula for the maximum sum M(A, j) of a contiguous subsequence ending on position j of an integer array A is:

M(A, j) = max(M(A, j – 1) + A[j], A[j])

Given the C++ template class below (from class)

```
template < class Arg, class Result >
class UnaryMemoFunction {
private:
  typedef std::map< Arg, Result > Cache;
  Cache cache;
public:
  Result operator() ( Arg a ) { return memo( a ); }
protected:
  Result memo( Arg a ) {
    typename Cache::const_iterator it = cache.find( a );
    return it == cache.end() ? cache[ a ] = call( a ): it->second;
  }
  virtual Result call( Arg n ) = 0;
};
```

fill in the code below to make the example test and ones like it pass:

```
class MaxSum …
{









};

TEST(MaxSum)
{
  int a1[] = { -2, 11, -4, 13, -5, -2 };
  std::vector<int> v( a1, a1 + 6 );
  // make a function object that sums over v
  MaxSum maxSummer( v );
  //find largest subsequence sum in v
  int result = -1;
  for ( unsigned int i = 0; i < v.size(); ++i)
    result = std::max( result, maxSummer( i ) );
  CHECK_EQUAL( 20, result );
}
```