Name: _____

ID Number: _____

UNIVERSITY OF VICTORIA
EXAMINATIONS- DECEMBER 2011
CSC 225- Algorithms and Data Structures: I
Section A01, CRN 10568
Instructor: Wendy Myrvold
Duration: 3 hours

TO BE ANSWERED ON THE PAPER.

Instructions:

This question paper has twelve pages (the last page is blank in case you need extra space) plus the header page.

Students **MUST** count the number of pages in this examination paper before beginning to write, and report any discrepancy immediately to the invigilator.

Use only space provided on exam for answering questions. Closed book. No aids permitted.

Question	Value	Mark
1	20	
2	10	
3	10	
4	10	
5	20	
6	10	
7	20	
Total	**100**	

1. Circle true or false for each question and justify your answer. **No marks will be given unless there is a correct justification.**

(a) [5 marks] Suppose singly linked lists are used to implement a queue. Then, adding to the queue should be done at the front of the list, and deletions from the queue should be taken from the end of the list.
True **False**

(b) [5 marks] An algorithm which takes $O(n^2)$ time in the worst case is always faster on worst case examples than an algorithm which is $O(n^3)$ in the worst case.
True **False**

(c) [5 marks] Let f, g, and h be functions from the natural numbers to the positive real numbers. Then if $g \in \Omega(f)$ and $g \in O(h)$, and $f \in O(h)$ then $g \in \Theta(h)$.
True **False**

(d) [5 marks] Hashing takes $O(1)$ time in the worst case to insert an item into a hash table.
True **False**

2.(a) [3 marks] State the defintion presented in class for Ω.

(b) [2 marks] Give a function $f(n)$ which is as simple as possible such that $n^2 - 200 \in \Theta(f(n))$.

(c) [5 marks] Use the definition from (a) to prove that $n^2 - 200 \in \Omega(f(n))$ for the $f(n)$ you chose for part (b).

3.(a) [8 marks] Give Java code for a recursive middleMin algorithm that resides in the LinkedList class as defined on our assignments. It should return a pointer, *prevPtr*, which is null if the minimum value is at the start of the list, and otherwise it points to the cell just before the one which has the minimum value. The algorithm should solve the problem by splitting the list in half, solving each half recursively, and then marrying the solutions.

public ListNode middleMin( ) // returns a pointer *prevPtr* to a list node.

(b) [2 marks] Set up (but do not solve) a recurrence for the time complexity of your middleMin algorithm from (a). Justify your answer in terms of what your method is doing.

4. Provide a trace of the Dijkstra/Prim minimum spanning tree algorithm (as described in class). Show the values of the data structures *tree*, *min_wt*, and *closest* after each phase of the algorithm. The *phase* equals the number of vertices in the tree so far.

(a) [1 mark] *tree:*

Phase:	0	1	2	3	4	5	6
a							
b							
c							
d							
e							
f							

(b) [4 marks] *min_wt:*

Phase:	0	1	2	3	4	5	6
a							
b							
c							
d							
e							
f							

(c) [4 marks] *closest:*

Phase:	0	1	2	3	4	5	6
a							
b							
c							
d							
e							
f							

(d) [1 mark] Mark the edges in the minimum weight spanning tree.

5. For this question, use weighted union (weight is the number of nodes in a component), with **collapsing** find. Consider this picture:

(a) [2 marks] What is stored in the parent array?

0	1	2	3	4	5	6	7	8	9

(b) [4 marks] Give the parent array that shows the result after a collapsing find, C-FIND(4) is called.

0	1	2	3	4	5	6	7	8	9

(c) [4 marks] Give the parent array that results from applying W-UNION(0,8) to your answer from part (b).

0	1	2	3	4	5	6	7	8	9

[Question #5, continued]

(d) [10 marks] Prove using induction that for weighted union (with the number of nodes in each component used as the weight) that:

Statement S(h): A tree of height *h*, formed by the policy of merging the tree with fewer nodes into the tree with a larger number of nodes has at least 2^h nodes.

6.(a) [8 marks] Perform DFS (Depth First Search) on the graph given below. Start at vertex 0. When traversing the neighbours of a vertex, traverse them **in numerical order**. Show all your work including the contents of the stack (at each step), the parent array and the DFI array.

	0	1	2	3	4	5	6	7	8
Parent:									
DFI:									

Stack contents at each phase:

(b) [2 marks] Mark the edges of the DFS tree on the picture.

7. For this question, refer to the program on page 11 (it is the same as the code I gave you on Assignment 2A). Consider the three reverse methods *beginReverse*, *middleReverse* and *endReverse* (*beginReverse*, and *endReverse* are described in a comment inside the *middleReverse* method).
For a list of size n, $B(n)$ will be used to denote the number of references to a **next** field for *beginReverse*, $M(n)$ will be used to denote the number of references to a **next** field for *middleReverse* and $E(n)$ will be used to denote the number of references to a **next** field for *endReverse*.
The program provided is divided into four blocks (A, B, C, and D as indicated on the code provided).

(a) [5 marks] For each block and algorithm variant, fill in this chart with the number of times that a **next** field of a ListNode is accessed for a problem of size n.

	beginReverse	*middleReverse*	*endReverse*
Block A			
Block B			
Block C			
Block D			

Use floor/ceiling functions in your answer as required.

(b) [5 marks] Use your table from (a) to set up recurrence relations for $B(n)$, $M(n)$ and $E(n)$.

[Question 7 continued]

(c) [5 marks] $M(n)$ is equal to the number of times that *middleReverse* accesses a **next** field of a *ListNode* object. Assume that $n = 2^k$ for some integer $k \geq 0$. Solve your recurrence for $M(n)$ from question 7(b) using repeated substitution to get a closed formula.

Your recurrence relation from (b):

Solve your recurrence relation:

[Question 7 continued]

(d) [5 marks] Assume that $n = 2^k$ for some integer $k \geq 0$. Prove by induction that your answer to 7(c) correctly gives the number of times a **next** field is accessed when reversing a list of size n for middleReverse.

Your closed formula from (c):

Inductive proof:

The program for Question 7:

```
  public void middleReverse(int level)
  {
     LinkedList listA, listB;
     int i, size;

//   Block A: Base case: a list of size 1 is the same reversed.
     if (n <= 1) return;

//   Block B: Split list into two lists where first one has size floor(n/2).
     listA= new LinkedList( );
     listB= new LinkedList( );

     size= n/2; // Variants have size=1 (beginReverse) or size= n-1 (endReverse)

     listA.n= size;
     listA.start= start;
     listA.rear= start;
     for (i=0; i < size-1; i++)
          listA.rear= listA.rear.next;

     listB.n= n-size;
     listB.start= listA.rear.next;
     listB.rear= rear;

//   Null terminate the first list.
     listA.rear.next= null;

//   Block C: Solve the two subproblems recursively.
     listA.reverse(level+1);
     listB.reverse(level+1);

//   Block D: Marry the two solutions.
//   The value for n is still correct.
     start= listB.start;
     rear = listA.rear;

//   Connect the two lists together.
     listB.rear.next= listA.start;
  }
```

Use this page if you need extra space. Clearly indicate the question you are answering.

The END.