Name: _____

ID Number: _____

UNIVERSITY OF VICTORIA

EXAMINATIONS- SUMMER 2001

CSC 225 F01

Instructor: W. Myrvold

Duration: 3 hours

TO BE ANSWERED ON THE PAPER.

Instructions:

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

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

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	20	
7	15	
Total	**105**	

1. [20] Circle true or false for each question and justify your answer. No marks will be given unless there is a correct justification.

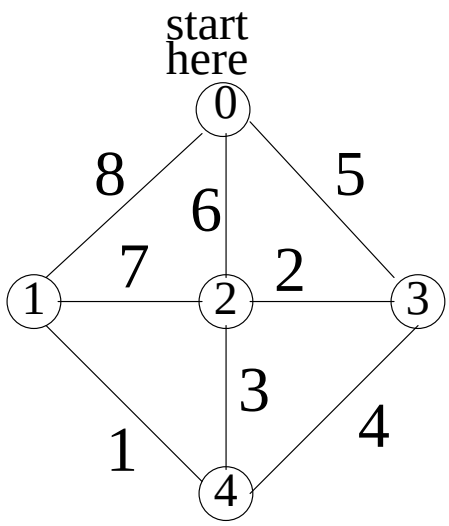
(a) It takes $\Omega(n \log n)$ time to build a heap of size n because this is how long it takes to do n insertions.
True **False**

(b) Radixsort which takes time $O(n)$ is faster on all inputs than any of the comparison models sorting algorithms we have studied (e.g. Heapsort, Mergesort, Quicksort, Maxsort).
True **False**

(c) If a Queue is implemented using a linked list then for efficiency reasons, insertions should be done at the beginning of the list and deletions should be done from the rear.
True **False**

(d) In order to find a minimum weight spanning tree, we might have to completely order the edges by weight. Thus under the comparison model, any algorithm for minimum spanning tree takes time at least $\Omega(m \log m)$ in the worst case where m is the number of edges.
True **False**

2. [10] Show the values of the data structures *tree*, *min_wt*, and *closest* (as described in class) after each phase of the Dijkstra/Prim minimum spanning tree algorithm. The *phase* equals the number of vertices in the tree so far. The columns are numbered by the phase and the rows by the vertex numbers. Mark the edges in the MST.



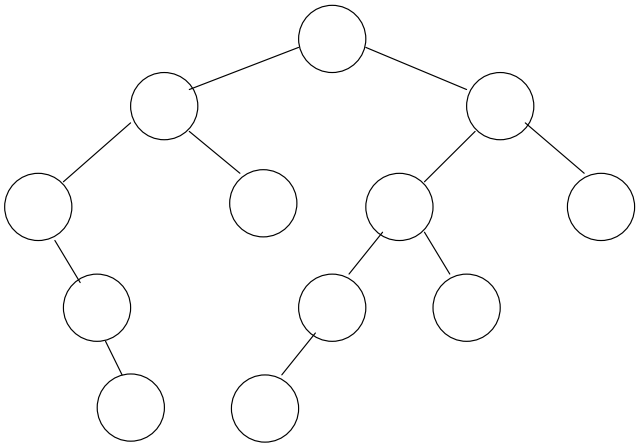
tree:	0	1	2	3	4	5
0						
1						
2						
3						
4						

min_wt:	0	1	2	3	4	5
0						
1						
2						
3						
4						

closest:	0	1	2	3	4	5
0						
1						
2						
3						
4						

3.(a) [3] Insert keys into the binary tree structure given so that a **in-order** traversal will give:

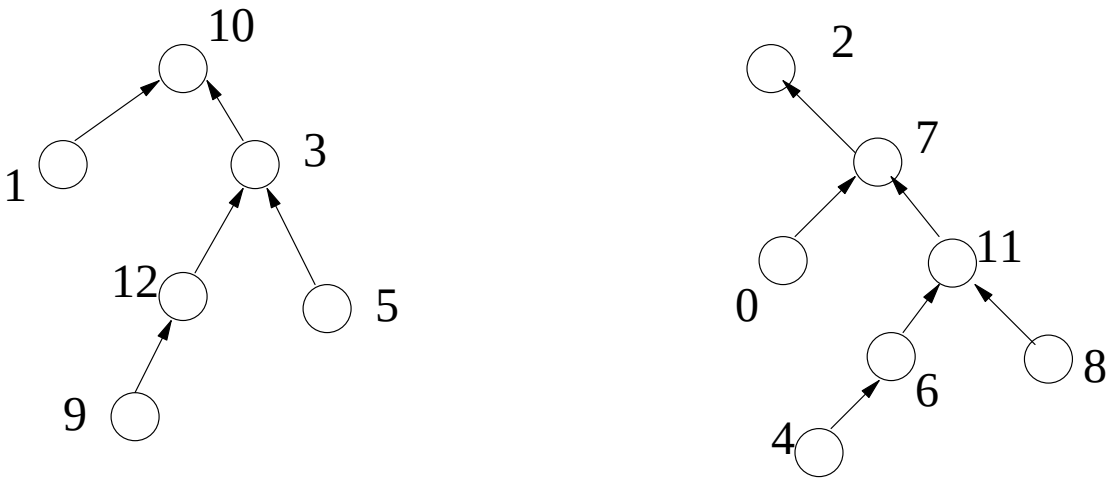
0	2	3	4	5	6	8	9	11	14	16	21



(b) [3] Give the psuedocode for a in-order traversal of a binary tree.

(c) [4] What is the time complexity of doing an in-order traversal of a binary tree having *n* nodes? Justify your answer.

4. Weighted UNION/Collapsing FIND



(a) [6] Show the updated data structure after UNION(9, 6) assuming that weighted UNION and collapsing FIND are used.

(b) [2] What does the parent array contain before the update?

0	1	2	3	4	5	6	7	8	9	10	11	12

(b) [2] What does the parent array contain after the update?

0	1	2	3	4	5	6	7	8	9	10	11	12

5. [20, -2 marks for each incorrect answer] Your job is to analyse various data structures for implementing a set. The sets under consideration can contain up to n elements represented by the integers from zero up to $n - 1$. Consider the following data structures for a set S:

   (a) The elements of set S are stored in a sorted array.

   (b) The elements of set S are stored in an unsorted linked list [the linked list should not contain any duplicate values].

   (c) The elements of S are indicated by a *characteristic vector* which is an array of size n which has position i set to 1 if element i is in the set and 0 if it is not in the set.

Let S be a set which has size s and let T be a set of size t. Give the **worst case** time complexities of algorithms for the following set operations as functions of n, s, and t.

Operation	(a) Sorted array	(b) Unsorted list	(c) Characteristic vector
Is x in S?			
Add x to S (do not count time to check if x in S)			
Print elements in S in sorted order			
Create union of S and T			

6. Suppose the *s* elements of set *S* [from Question 5] are stored in an unsorted array and you are restricted to an algorithm which is valid under the comparison model.

   (a) [10] Give the pseudo code for an algorithm for traversing the set elements in sorted order which has optimal worst case time complexity.

   (b) [10] Analyze the worst case time complexity of your algorithm from (a) using Big Oh notation.

7. Consider the following divide and conquer approach for reversing the order of the elements on a linked list.

   reverse_order(n, start, end)

   Input: n- the number of items in the list.
   start- a pointer to the start of the list.

   Output: start- points to the first cell of a new list having the keys in the reverse order as the original list.
   end- a pointer to the last cell on the list.

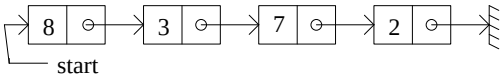
The divide and conquer strategy you must implement to accomplish this is as follows:

1. Divide the list into two sublists L1 containing the first $\left\lfloor \frac{n}{2} \right\rfloor$ items and L2 which contains the remaining items.
2. Reverse L1 and L2 recursively.
3. Connect the two reversed lists together to get a reversal of the original list.

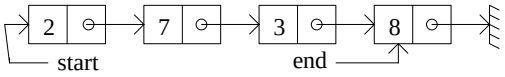
(a) [5] What is the worst case time complexity of this algorithm? Justify and solve an appropriate recurrence relation.

(b) [10] Give detailed pseudo code (almost JAVA code but without worrying about syntax) for the *reverse_order(n, start, end)* routine described on the previous page. To get full marks, your solution must implement the algorithm as specified.

If the initial list is:



On termination, the list is:



reverse_order(n, start, end)

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

The END.