BROCK UNIVERSITY

Final Examination:	Fall 2009	Number of Pages: 7
Course:	COSC 2P03	Number of students: 30
Date of Examination:	18th December 2009	Number of Hours: 3
Time of Examination:	9:00 - 12:00	Instructor: S. Houghten

Instructions:

1. A minimum of 40% must be obtained on this examination in order to achieve a passing grade in the course.
2. All questions are to be answered in the exam booklets provided.
3. No electronic devices are permitted.
4. No examination aids are permitted.
5. Use or possession of unauthorized materials will automatically result in the award of a zero grade for this examination.
6. This examination has a total of 90 marks.

Question 1 (16 Marks) – Short Answer
Write short but complete answers to each of the following. Use examples as necessary to demonstrate your point.

a. **[2x2 marks]** Explain the motivation for maintaining a *height-balanced tree*. Suppose that we require that the left subtree and right subtree of the root differ in height by at most 1: explain why this balance condition is/is not useful.

b. **[2 marks]** What two properties must be satisfied by a binary tree if it is a *min-heap*?

c. **[2x2 marks]** Suppose that we have an undirected graph. What two properties must this graph hold if it has an *Euler circuit*? In each case, clearly explain why the graph must have the property.

d. **[3 marks]** Given a directed graph G, we wish to create a list of the vertices of G that is in *topological order*. When is it **not** possible to produce such a list? Give an example.

e. **[3 marks]** Give an example of a structure that allows for good search time at the cost of using additional space, and clearly explain how the additional space reduces search time.

Question 2 (11 Marks) – Complexity and Recursion

a. **[3 marks]** Express the complexity of the following method using big-O notation. You must explain how you arrived at your answer. What will be printed if the initial value of n is 16?

```
hohoho(int n)
{
      print("Ho");
      if(n == 0)
            print("Happy Holidays");
      else if(n < 5)
            hohoho(n-1);
      else
            hohoho(n/2);
}
```

b. **[8 marks]** Suppose that we have an undirected graph with |V| vertices and |E| edges.

(i) [3 marks] What is the complexity of breadth-first search if we use an *adjacency matrix*? You must explain your answer.

(ii) [3 marks] What is the complexity of breadth-first search if we use an *adjacency list*? You must explain your answer.

(iii)[2 marks] If the graph is *sparse*, which representation is preferable, and why?

Question 3 (11 Marks) – Trees and Traversals

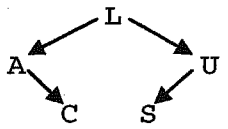
a. **[5 marks]** Suppose that we have a general tree in which the nodes have the following representation:

```
class TreeNode
{
  int contents;
  TreeNode firstChild;
  TreeNode nextSibling;
}
```

Write a recursive method `postOrder(TreeNode n)` that will perform a *postorder* traversal of the tree with root `n`.

b. **[3 marks]** Consider a binary tree structure in which each node stores a single character. This tree is *not* necessarily a binary search tree. Is it possible to create such a tree if the tree has inorder traversal c,a,b,d and preorder traversal a,b,c,d? If it is possible, draw the tree; if it is not possible, explain why. **Hint:** as a first step, determine which node must be the root.

c. **[3 marks]** Draw the following tree with successor threads only, when the tree is threaded for *postorder* traversal. Is it possible to use these threads for postorder traversal of the tree? Why or why not?



Question 4 (11 marks) – AVL Trees and B-trees

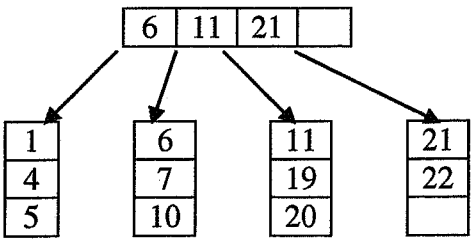
a. [2 marks] Write a method

```
AVLNode doubleWithLeftChild(AVLNode k3)
```

that will restore the AVL property to a tree in which an insertion into the right subtree of the left child of `k3` caused an imbalance at `k3`. The method should return the root of the modified tree. You may make use of (but do not need to write) the following methods:

```
AVLNode rotateWithLeftChild(AVLNode k2)
      // single rotation from left→right on k2,
      // returning root of modified tree
AVLNode rotateWithRightChild(AVLNode k2)
      // single rotation from right→left on k2,
      // returning root of modified tree
```

b. [2x2 marks] Suppose that we have the following B-tree of order 5, in which each leaf node can store a maximum of 3 records. Note that for simplicity, only the keys of the records are shown. Draw the tree after each of the following steps is performed in turn:



(i) Insert 2

(ii) Insert 9 (into the tree resulting from step (i) above)

c. [5 marks] Suppose we wish to create a B tree on a computer with a block size of 512 bytes and pointers of 8 bytes. The records we wish to store in the tree are 50 bytes each, **including** keys of 12 bytes.

(i) [2 marks] What is the *order* of the tree?

(ii) [2 marks] What are the *minimum* and *maximum* numbers of records that can be stored in a leaf node that is not the root?

(iii)[1 mark] What is the minimum number of children for a root node that is not a leaf?

Question 5 (11 Marks) – Sorting

a. [5 marks] Write a method `heapsort(int [] A, int n)` to perform Heapsort using a *max-heap* on the elements of `A`.

You may assume that the elements to be sorted are stored in `A[1..n]` and that `A[0]` contains a sentinel value larger than any value in `A[1..n]`.

You may also assume the existence of the following methods:
`trickleUp(int [] A, int i, int n)`: trickle up from index `i`, in heap of size `n`
`trickleDown(int [] A, int i, int n)`: trickle down from index `i`, in heap of size `n`
`swap(int [] A, int i, int j)`: in A, swap the item at index i with the item at index j

You may *not* assume that A[1..n] is already a max-heap before calling `heapsort`.
Upon exit from `heapsort`, the elements of `A[1..n]` should be in increasing order.

b. [3x2 marks] For each of the 3 separate scenarios below, which sorting algorithm would you choose, if you were given a choice of using Mergesort, Quicksort, Heapsort, or Radix Sort? Note that in each case, there may be more than one suitable answer, so you **must** explain your choice.

(i) You must minimize the amount of space used.

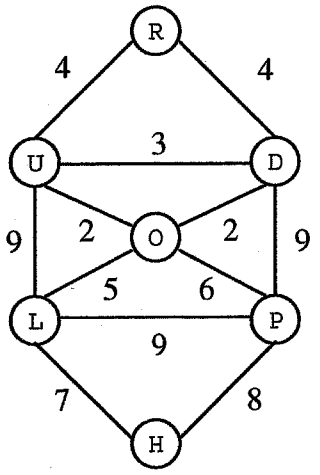
(ii) 1 billion numbers are to be sorted, each of which has a value between 0 and 777.

(iii) You know that there are long sequences within the array which are already sorted.

Question 6 (12 Marks) – Graphs

a. **[3 marks]** When using Dijkstra's algorithm for finding shortest paths in a weighted graph, we can use a table in which one column indicates whether a given vertex is "known". Once a vertex v is declared "known", this indicates that we know the cheapest possible path from the source vertex to v. Why is not possible to find a shorter path to v after it is declared "known"? **Hint:** remember that several other vertices still might not have been examined – think about what it means if a vertex has yet to be examined.

b. **[9 marks]** Use *Kruskal*'s algorithm to find a minimum spanning tree (MST) of the following graph. As each edge is considered, indicate whether that edge is or is not chosen, and why. Draw the MST once it is complete.



Question 7 (18 Marks) – Hashing

a. [2x2 marks] Suppose that you are using a hash table of size `tableSize` with a hash function `h(key) = key % tableSize`. Briefly explain two problems that can occur if you are using open addressing and `tableSize` is *not* a prime number.

b. [3x2 marks] Suppose that you are using a hash table with open addressing.

(i) What is primary clustering and how does it occur?

(ii) What is secondary clustering and how does it occur?

(iii)How does double hashing avoid both primary and secondary clustering?

c. [8 marks] Suppose that we have a set of records with the following keys:

```
36, 38, 7, 4, 19, 21, 17, 6.
```

Suppose that these records are inserted one at a time, in the order given, into an extendible hash structure. This structure is initially empty with global depth 1 (and thus, has 2 empty buckets each with local depth 1). Each bucket can hold a maximum of 2 records, and we are using the hash function `h(key) = key mod 12`.
Show the status of the extendible hash structure after *every* insertion.

Bonus Question (1 mark): What is the name of the easiest type of deletion?