University of Toronto
Department of Electrical and Computer Engineering

Final Examination

ECE-242 Algorithms and Data Structures
Spring Semester, 2001

2:00pm–4:30pm, Thursday, April 26, 2001

Print your name and your student number neatly in the space provided below; print your student number at the upper right corner of every page. Place a picture ID on your table for verification.

Last Name:	
First Name:	
Student Number:	

This booklet should be twelve pages long including this page. If not, report this to the proctor. Do all problems in this booklet. Try not to spend too much time on any one question. *When the question requires you to provide code fragments as an answer, the less code you write the more credit you will receive. We assume that you use the notation presented in lectures. If you use other notation, you will need to state your assumption(s).*

This is a closed book/lecture notes exam. Calculators are not allowed. Do all work in the space provided. You can use the back of the sheets as scrap paper, if necessary. Ask the proctor if you need more paper. **Nothing on the back of the sheets will be graded. Write your answers clearly: you may not receive full credit if your answer is unreadable.**

Question	Points	Score	Grader
Multiple Choice	44		
Short Answers	66		
Total	110		

(Multiple choice, 44 Points) Each of the multiple choice questions is worth 2 points and it has *only one* correct answer. If we cannot understand which answer you circled you will not get credit for the question. If you make any assumption(s), state it clearly.

1. Consider an array $A[1 \ldots n]$ that stores a heap with n distinct elements where the maximum element is at the top. Where can the minimum element be located? Choose the best answer.

   a) $A[2..n]$ b) $A[n]$ c) $A[n-1] or A[n]$ d) $A[2]$ or $A[4]$ or $A[6]$ e) $A[(n+1)/2..n]$

2. Using open addressing and double hashing in a hash table with m slots, where m is a prime number, what is the average number of probes to insert an element if the table has exactly one empty slot?

   a) m b) $m-1$ c) $m/2$ d) 1 e) none of the above

3. Which traversal gives a decreasing order of elements in a heap where the max element is stored at the top?

   a) pre-order b) in-order c) post-order d) level-order e) none of the above

4. AVL tree rotations

   a) are performed on every insertion

   b) are used to readjust the levels of the tree to have a height difference of 2

   c) take worst case $O(n)$ time

   d) are used to maintain a worst case search time of $O(\log n)$

   e) none of the above

5. If a value sitting in a heap has its key decreased, how long it should take to readjust the heap in the *worst case*?

   a) $O(n \log n)$ b) $O(n)$ c) $O(\log n)$ d) $O(n^2)$ e) $O(1)$

6. Consider the following sequence of push operations in a stack:

```
push(S,'1');
push(S,'2');
push(S,'3');
push(S,'4');
push(S,'5');
push(S,'6');
```

You can insert as many `pop(S)`'s as you like in the above sequence to get a desired output. Which of the following cannot be an output?

a) 123456 b) 325416 c) 342561 d) 342615 e) 342165

7. Which of the following does *not* describe a difference between singly-linked lists, and stack and queues?

a) a list allows access to all of its elements at any given time, whereas stacks and queues only allow access to one

b) you can insert into any position in a list, while you can only insert into one position in a stack or a queue

c) removing from a list can sometimes take $O(n)$ but `Pop()` and `Dequeue()` always take $O(1)$ time

d) inserting into a list is always $O(n)$ but you can insert into a stack or queue in $O(1)$ time

e) none of the above

8. What is the *worst case* complexity to insert into a queue holding n items, if the queue is implemented with head and tail pointers in an array with wraparound?

a) $O(1)$ b) $O(\log n)$ c) $O(n)$ d) $O(n^2)$ e) $O(n \log n)$

9. What is the running time of breadth-first search when an adjacency list is used to implement a graph?

a) $O(|V|)$ b) $O(|V| * |E|)$ c) $O(|V| + |E|)$ d) $O(|V|^2)$ e) $O(|E|)$

For questions 10 and 11, let $G = (V, E)$ be a graph represented by an adjacency list where each edge $(m, n) \in E$ is represented *only once*: it can be found in the list for node labeled min(m, n). Let also deg(v) be the degree of vertex v.

10. Which of the following best describes the *worst case* runtime complexity to determine if nodes v and w are adjacent?

    a) O(1) b) O($|V|$) c) O(min(deg(v), deg(w)))

    d) O($|E|$) e) O(max(deg(v), deg(w)))

11. Which of the following best describes the *worst case* runtime complexity to list all nodes adjacent to v?

    a) $|V|$ b) $|E|$ c) deg(v) d) $|V|$deg(v) e) $|V||E|$

12. Increasing the order of a given B-Tree *cannot* ever:

    a) reduce the tree's height

    b) increase the maximum number of indices stored in an internal node

    c) increase the minimum number of indices stored in an internal node

    d) change the number of leaves in the tree

    e) none of the above

13. Let heap stored in an array as A = [3, 8, 4, 9, 11, 5, 6, 10, 12, 13]. Assume that the index of the first array cell is 1 and that this is also the root of the heap (*i.e.*, element 3). In other words, the root of the heap contains the minimum element. What is the result of inserting 7 into this heap?

    a) [3, 7, 4, 9, 8, 5, 6, 10, 12, 13, 11] b) [3, 4, 7, 9, 8, 5, 6, 10, 12, 13, 11]

    c) [3, 7, 4, 9, 8, 5, 6, 10, 12, 11, 13] d) [3, 4, 7, 9, 8, 5, 6, 10, 12, 11, 13]

    e) [3, 8, 7, 9, 4, 5, 6, 10, 12, 11, 13]

14. What is the *worst case* run time for heapsort?

    a) O($\log n$) b) O($n \log n$) c) O(n) d) O(1) e) O(n^2)

15. Suppose that you have an algorithm that returns in O(n) time the $\lceil \frac{15n}{16} \rceil$-th largest number from a list of n unsorted distinct integers. If you use this algorithm to chose a pivot then Quicksort takes *worst case* time

    a) O($\log 15/16n$) b) O($n \log n$) c) O(n) d) O($n^{15/16}$) e) O(n^2)

16. Which of the following is *not* true of Dijkstra's algorithm?

    a) the length of the shortest path to the start vertex is always zero

    b) it can find the shortest paths to *all* other vertices in the same worst case time that it needs to find the shortest path to a single vertex

    c) it will work on any weighted graph with positive weights

    d) it can be implemented to run in $O(V)$ time

    e) it takes time polynomial to the number of vertices

17. Which one of the following data structures can support the `Insert()` operation in constant time even in the worst case?

    a) Heap b) AVL Tree c) B-Tree

    d) Hash Table with Open Addressing e) Hash Table with Chaining

18. What is the average insert time complexity for a heavily-loaded hash table with chaining that contains n elements?

    a) O($\log n$) b) O(1) c) O(n) d) O($n \log n$) e) O($\frac{n+1}{2}$)

19. Which is the *worst case* running time of depth-first search assuming the graph is represented with an adjacency list?

    a) $O(|V| + |E|)$ b) $O(|V| \log |E|)$ c) $O(|V||E|)$ d) $O(|V|^2)$ e) $O(|V|^2|E|)$

20. Which of the following traversals on a Binary Search Tree will produce the same result regardless of the order in which the values were initially inserted to the tree?

    a) Preorder b) Postorder c) Levelorder d) Inorder e) a + b

21. An AVL tree requires

    a) that every node's subtree heights are equal

    b) several rotations to complete each insertion

    c) constant time to perform a rotation on it

    d) that every node's key is less than the keys in that node's children

    e) that the tree is a minimum-level binary tree

22. What is the average insert time complexity for a sparsely-loaded hash table with chaining that contains n elements?

    a) $O(\log n)$ b) $O(1)$ c) $O(n)$ d) $O(n \log n)$ e) $O(n^{1-a})$

(Short Questions, 66 points)

1. For each part of the problem. *name* and *justify* one data structure that addresses the problem. Your answer should not be more than 3 lines. You may choose from the following list of data structures:

Stack Queue Binary search tree AVL tree Heap Hash table

(a) The data structure is initially empty. We then insert the values 2. 10. 12. 8. 6 and 4. in that order. Now. the only element we can remove is 12. (3 points)

(b) The data structure is initially is empty. We insert the values 2. 4. 6. 8. 10. 1 and 7. in that order. If we want to remove an element. our only choice is 2 (3 points)

(c) The data structure initially contains n elements. We then insert elements 7. 14. 27. 68. and 3. We may now find any element in average case O(1) time (3 points)

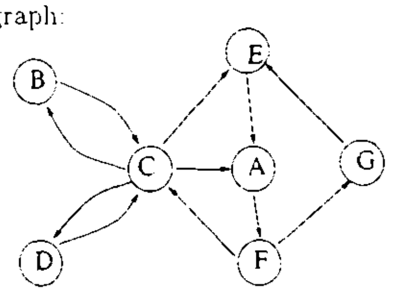
(d) The data structure initially contains n elements. We then insert elements 2. 7. 5, 13. 11. 3 and 1. in that order. We may insert any element in worst case $O(\log n)$ time (3 points)

2. Consider the following C function:

```
int f(int n) {
  if (n == 0) {
    return 0;
  } else if (n%10 == 0) {
    return f(n/10);
  } else {
    return 1 + f(n-1);
  }
}
```

Give a *short* and *simple* description of what `f(n)` returns when $n \geq 0$. To receive credit, your answer should *not* be more than 3 lines long (5 points). What is the *worst case* time complexity (Big-Oh) of the function in terms of n? Justify your answer in 2 lines or less. (2 points)

3. Consider the following directed graph:



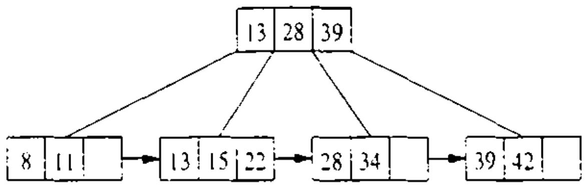
Draw *two different* trees that can be generated by depth-first traversals starting at A. Do *not* draw the same tree twice with different ordering of the children because these trees are considered the same (4 points).

4. The following is a preorder traversal of a binary search tree

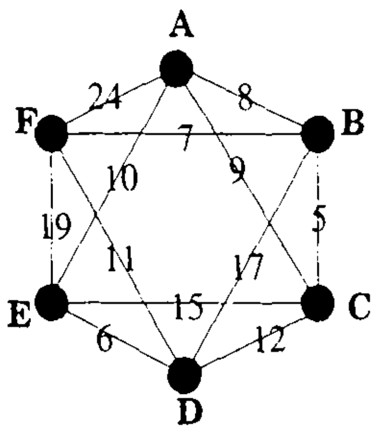
PREORDER: 20 10 6 5 15 26 22 24 23

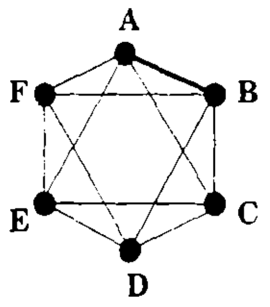
Draw the binary search tree (5 points)

5. Given the B+-tree below, draw the *final* tree that results after we insert 18 (6 points).

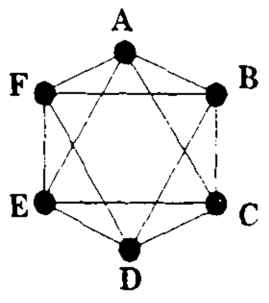


6. Given the graph below, produce the shortest paths from starting vertex A using Dijkstra's algorithm. Draw in boldface the edges selected at each step of the algorithm using the smaller pictures. Let the indices in these figures correspond to consecutive steps of the algorithm. We have drawn the first step as an example. The bold edge means that it has been selected (5 points). For every step, you will also need to fill the boxes with a number that indicates the length of the shortest path from A to each vertex through vertices that have been already discovered. Again, we give you the the first step as an example (5 points).

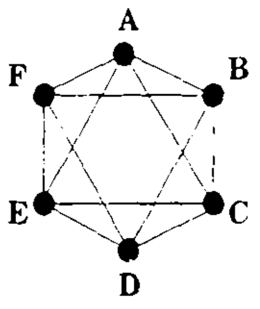




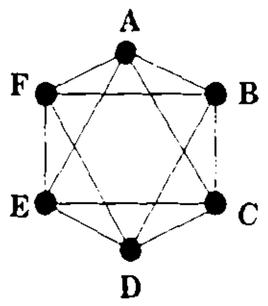
a)



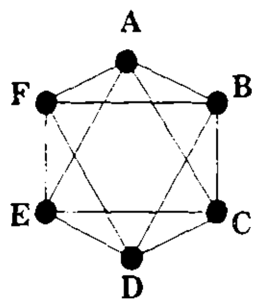
b)



c)



d)



e)

Step a)	A [0]	B [8]	C [9]	D [ ]	E [10]	F [24]
Step b)	A [ ]	B [ ]	C [ ]	D [ ]	E [ ]	F [ ]
Step c)	A [ ]	B [ ]	C [ ]	D [ ]	E [ ]	F [ ]
Step d)	A [ ]	B [ ]	C [ ]	D [ ]	E [ ]	F [ ]
Step e)	A [ ]	B [ ]	C [ ]	D [ ]	E [ ]	F [ ]

7. Complete the following C function `contains()` using *recursion*. The function takes an array `arr` of integers sorted in increasing order, the length `n` of this array and an integer `v` to search for. The function returns 1 if v is in the array, 0 otherwise. To get full credit, the worst case complexity of the function must be $O(\log n)$. You can assume that we always pass for n a number greater or equal to 0 (7 points).

```
int contains(int* arr, int n, int v) {









}
```

8. A node with key X is deleted from a *Binary Search Tree*, and then the same key X is immediately re-inserted into the tree. Assume that deletion and insertion are done using any of the algorithms presented in lectures notes. If the original tree (before deletion and insertion) and the final tree (after deletion and insertion) are identical, then some conclusions can be drawn. For each of the four claims below, place an "X" under one of the columns, to indicate whether that claim is always true, never true, or indeterminate (6 points).

CLAIM	ALWAYS	NEVER	CANNOT TELL
The node involved is the root			
The node involved is a leaf			
The tree has at least one level			
It is an AVL tree			

9. A student who slept through most ECE-242 lectures wrote the following code for Lab 7. It contains several compile time errors and/or run-time errors (bugs). Point out all bugs by circling the relevant code and placing a label besides that circle. Make a list of these bugs in the space below and describe how to fix the error. We give you the first such footnote as an example. If you don't recall the details of Lab 7, its description can be found as a comment. (9 points).

```
#include <stdio.h>

/* Max char count in line buffer */
#define LINE_CHAR_LIMIT 1024;

/* Reads from stdin until either an EOF condition or newline character
   is encountered, or LINE_CHAR_LIMIT characters have been read.  The
   newline is read and stored as a character, and counts towards the
   LINE_CHAR_LIMIT.  The 'line_ptr' argument is set to point to a
   string buffer containing the line read, and the number of
   characters read is returned. */

int getline( char **line_ptr )
{
  char c, line[ LINE_CHAR_LIMIT ];
  int i;

  for( i = 0;
       i < LINE_CHAR_LIMIT && ( c = getchar() ) != EOF && c != '\n';
       ++i ) {
    line[ i ] = c 1
  }

  if( c == '\n' ) {
    line[ i ] = c;
  }

  line[ i ] = '0';

  line_ptr = line;

  return i;
}
```

1: Semicolon is missing at the end of line. Add one.