University of Toronto
Faculty of Applied Science and Engineering

CSC 190S

COMPUTER ALGORITHMS, DATA STRUCTURES AND LANGUAGES

Spring 1996

FINAL EXAM

Examiner: T.S. Abdelrahman

This test is open book(s) and open notes. Use of computing devices is NOT permitted.

Do not remove any sheets from this test book. Answer all questions in the space provided. No additional sheets are permitted.

Work independently. The value of each part of each question is indicated. The total value of all questions is 100.

Write your name and student number in the space below. Do the same on the top of each sheet of this test book.

Name (print): ______________________
(underline last name)

Student Number: ______________________

Q1.________	Q6.________
Q2.________	Q7.________
Q3.________	Q8.________
Q4.________	Q9.________
Q5.________	Q10.________

Total: $\boxed{}$

Question 1 (12 marks). *General.*

Answer the following questions by circling either **True** or **False**, or by providing a *brief* answer where appropriate.

1. The O notation we use in class to express the time complexity of an algorithm is useless in practice because it does not give actual execution time. HeapSort and QuickSort are a case in point. Although both have average time complexity of $O(N log N)$, QuickSort is faster and is more common in practice.

   True or **False**?

2. Although quadratic probing eliminates both primary and secondary clustering, it does not probe all the hash table locations, hence, underutilizing the table.

   True or **False**?

3. Heuristic search reduces the worst-case time complexity of searching for solutions from exponential to polynomial.

   True or **False**?

4. A stack is used to maintain the list of states in depth-first heuristic search.

   True or **False**?

5. The depth-first strategy is always better than the breadth-first strategy in heuristic search algorithms.

   True or **False**?

6. The set of all valid sentences in a language is the set T^*, where T is the set of terminals.

   True or **False**?

7. What is wrong with the following hashing function, expressed in some language?

```
/* returns index value */
int index (int key /* key to be hashed */) {
    count = count + 1; /* count is a global variable */
    index = (key mod count) + 1;
}
```

8. What is the purpose of AVL trees?

9. In one sentence, explain how the heapsort program you implemented in assignment 5 can be modified to sort the array in decreasing order instead of increasing order.

10. An array sorted in decreasing order is a valid heap.

True or False?

11. Can the hashing function used in assignment 6 be made better by using a *truly* random number generator? Explain briefly.

12. When would it be better to use a binary search tree and not a hash table to search for a key in a collection of keys?

Question 2 (10 marks). *Complexity Analysis.*

a. **(5 marks)** Determine the time complexity of the following program segment as a function of n, the input size.

```
w=0
for i : 1 .. n
  for j : 1 .. n*n
    for k : 1 .. n*n*n
       w = w + 1
    end for
  end for
end for
```

b. **(5 marks)** Determine the time complexity of the program segment shown below as a function of the size of the input n. Show the details of your analysis and clearly indicate your final result. Assume for simplicity that n is a power of two.

```
function recursive (n : int) : int

   var x,y : int
   if n <= 1
      then result 0
      else for i:1 .. n
              O(1)
           end for
           x:=recursive (n div 2)
           y:=recursive (n div 2)
           result x+y
   end if
 end recursive
```

Question 3 (8 marks). *Tree Traversals.*

a. **(5 marks).** Given the following inorder (LNR) and postorder (LRN) traversals of a binary tree, draw the tree.

Inorder: C B A E D F H I G K J
Postorder: A B C D E F I K J G H

b. **(3 marks).** Given the following preorder (NLR) and postorder (LRN) traversals of a binary tree, is it possible to construct the tree uniquely? If possible, then give the tree; if not then show two different trees that have the traversals given below.

Preorder: A B C D
Postorder: D C B A

Question 4 (15 marks). *Heaps.*

The following is a 15 element array.

13	50	7	8	25	18	9	2	5	1	3	6	12	31	10

a. **(1 mark).** Draw the binary tree represented by the array.

b. **(3 marks).** Build a heap from the elements of the array. Show the tree represented by the array after each step of the process. If the tree does not change from one step to the next do not redraw the tree.

c. **(3 marks).** Start with the heap in part **b.** and insert a new element with the key "45" and restore the heap properties if necessary. Show the tree after each step of the process.

d. **(3 marks).** Start with the heap in part **b.** and delete the element with the largest key then restore the heap properties as needed. Show the tree after each step of the process.

e. **(5 marks).** Write a procedure to delete the item with index i from the heap and restore heap properties. You may find the heap procedures `BuildHeap`, `BubbleUp` and `BubbleDown` defined in class or in your textbook useful. Your solution should be around 6-7 lines of code. Use C or pseudocode.

Question 5 (13 marks). *Hash Tables.*

Assume a hash table with size $T = 13$, and the following hash function

index = (key **mod** T) + 1

a. **(5 marks)** Show the contents of the hash table after the following operations have been performed. Indicate next to each operation the number of *probes* performed. Assume that linear probing is used to resolve collisions and that the hash table is initially empty.

1. insert 34.
2. insert 22.
3. search 11.
4. insert 8.
5. insert 10.
6. search 11.

index	key
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	

b. **(4 marks)** Redo part **a.**, but using quadratic probing to resolve collisions. Again, indicate next to each operation the number of *probes* performed and assume that the hash table is initially empty.

1. insert 34.
2. insert 22.
3. search 11.
4. insert 8.
5. insert 10.
6. search 11.

index	key
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	

c. (4 **marks**) Redo part **a.**, but using double hashing to resolve collisions. Use the following function as the second hashing function

index = (key **mod** (T-2)) + 1

If collisions persist, resolve them using linear probing. Again, indicate next to each operation the number of *probes* performed and assume that the hash table is initially empty.

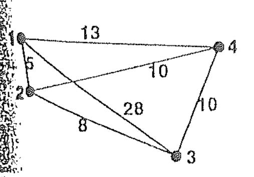
index	key
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	

1. insert 34.
2. insert 22.
3. search 11.
4. insert 8.
5. insert 10.
6. search 11.

Question 6 (10 marks). *Heuristic Search.*

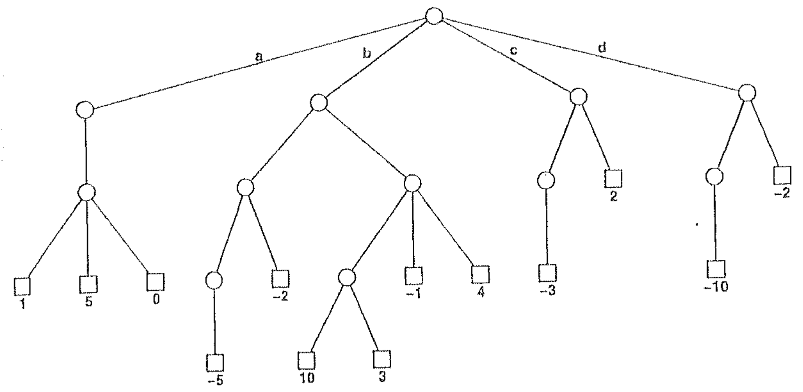
a. (5 **marks**). Show the search tree generated by breadth-first search to determine the satisfiability of the boolean expression $x_1\bar{x}_2 + x_1x_2\bar{x}_3$. State any assumption you make very clearly.

b. (5 **marks**). Show the search tree generated by depth-first search to find the optimal tour in the following travelling salesman problem. State any assumptions you make very clearly.



Question 7 (8 marks). *Game Trees.*

Part of the tree for a fictitious game is shown below. The value returned by the evaluation function applied to each leaf of the tree is indicated next to the leaf.



a. (4 **marks**). Using the minmax algorithm, indicate which move should be made by the machine (a, b, c or d).

b. (4 **marks**). Circle the parts of the tree that would be pruned had alpha-beta search been used instead. Assume the search is done depth-first left-to-right. Which move should be made by the machine in this case?

Question 8 (10 marks). *Grammars.*

Let $G = (T, N, S, P)$ be the following grammar:

$$\begin{aligned} T &= \{0,1\} \\ N &= \{Z, A\} \\ S &= Z \\ P &: Z \longrightarrow A \\ &\quad A \longrightarrow A0A \\ &\quad A \longrightarrow 1 \end{aligned}$$

a. (3 marks). Show that the sentence 10101 is valid in the language $L(G)$ generated by the grammar G. Show the syntax tree for the sentence.

b. (3 marks). Is the grammar G described above unambiguous? Justify your answer.

c. (4 marks). Write down a grammar for the language whose sentences consist of zero or more a's, followed by one b, followed by one or more c's.

Question 9 (6 marks). *Parsers.*

Let $G = (T, N, S, P)$ be the following grammar:

$$\begin{aligned} T &= \{0,1,2\} \\ N &= \{S\} \\ P &: S \longrightarrow 0S|1S|2 \end{aligned}$$

Write a recursive-descent paring function `parse_S` for the language $L(G)$ generated by the grammar G. You may assume two functions `current_token` and `next_token`, as defined in class, are available.

Question 10 (8 marks). *Data Structures.*

An abstract data type `SET` is to be implemented such that `SET` consists of a collection of data elements, with the following operations possible on `SET`:

- `INSERT E`: adds a new element `E` to `SET` only if it does not exit in `SET`.
- `DELETE E`: deletes the element `E` from `SET` provided it already exits in `SET`.
- `MEMBER E`: determines if element `E` exists in `SET`.
- `MAXIMUM E`: assigns `E` the largest element in `SET`.
- `MINIMUM E`: assigns `E` the smallest element in `SET`.

Seven possible data structures are being considered for `SET`, as listed in the first column of the table below. Determine the *average* time complexity for each operation on each data organization, and summarize in the following table. If an operation is not valid for a particular organization, enter "N/A" in the table. The structures are listed in no particular order.

data organization	Average Complexity (using O notation)				
	INSERT	DELETE	MEMBER	MAXIMUM	MINIMUM
Binary Search Tree					
Unordered Doubly Linked List					
Heap					
Hash Table					
Ordered Single Linked List					
Queue					
Stack					