UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

FINAL EXAMINATION, APRIL 2015

CSC 190 H1S — COMPUTER ALGORITHMS AND DATA STRUCTURES

Exam Type: C — NO calculator allowed
Examiner(s): Pirathayini Srikantha

Student Number: |_|_|_|_|_|_|_|_|_|_|_|_|_|

Family Name(s): _____

Given Name(s): _____

Lecture Section: ☐ LEC 01 (Mon 11 am - 12 pm) ☐ LEC 02 (Mon 5 pm - 6 pm)

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully*.

This final examination paper consists of 5 questions on 22 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy of the final examination is complete and fill in the identification section above.*

Answer each question directly on the examination paper, in the space provided. If you need more space for one of your solutions, use one of the extra "blank" pages at the end of the examination and *indicate clearly the part of your work that should be marked.*

Write up your solutions carefully! In particular, use notation and terminology correctly and explain what you are trying to do—part marks *will* be given for showing that you know the general structure of an answer, even if your solution is incomplete.

When writing code, comments are not required except where indicated, although they may help us mark your answers. They may also get you part marks if you can't figure out how to write the code.

MARKING GUIDE

# 1:	_____/ 30
# 2:	_____/ 15
# 3:	_____/ 15
# 4:	_____/ 15
# 5:	_____/ 25
TOTAL:	_____/100

Good Luck!

OVER...

Question 1. [30 MARKS]

This section consists of 20 multiple choice questions. For each question, circle **one** correct answer.

Part (a) [1.5 MARK] Which one of the following is true of trees?

(a) Trees can have simple cycles

(b) At most one node in a tree must be unconnected

(c) There must exist a unique path from the root to each leaf in the tree

(d) Not all leafs in an extended tree have NULL children

Part (b) [1.5 MARK] Which one of the following is true about various structures of a binary tree containing n nodes?

(a) Complete binary trees are dense and heights of these trees are typically in the order of n

(b) It is desirable to minimize the height of a binary tree so that the length of the path from the root to any node in the tree will not be too long

(c) The longest path in a right/left linear or zig-zag binary tree is $\lfloor log_2(n) \rfloor$

Part (c) [1.5 MARK] Which one of the following is a correct definition of a binary tree?

(a) A binary tree is either an empty tree or consists of nodes with left and right subtrees that are binary

(b) A binary tree consists of internal nodes that have two non-null children except for the leaf nodes

(c) A binary tree consists of nodes that can have from 0 to 2 children with the exception of the root node

Part (d) [1.5 MARK] What is the complexity of finding the second largest element in a min heap?

(a) $O(n)$

(b) $O(1)$

(c) $O(n^2)$

(d) $O(log(n))$

(e) $O(nlog(n))$

Part (e) [1.5 MARK] What is the complexity of finding the largest element in an AVL tree?

(a) $O(n)$

(b) $O(1)$

(c) $O(n^2)$

(d) $O(log(n))$

(e) $O(nlog(n))$

Part (f) [1.5 MARK] What is the worst case complexity of applying quick sort on the sequential implementation of max-heap where the pivot is selected to be the first element in each sub-array division?

(a) $O(n)$

(b) $O(1)$

(c) $O(n^2)$

(d) $O(log(n))$

(e) $O(nlog(n))$

Part (g) [1.5 MARK] What is the complexity of retrieving the smallest value in a hash table?

(a) $O(n)$

(b) $O(1)$

(c) $O(n^2)$

(d) $O(log(n))$

(e) $O(nlog(n))$

Part (h) [1.5 MARK] Which one of the following sorting algorithms performs better when applied to an array of sorted elements?

(a) Bubble sort

(b) Merge sort

(c) Heap sort

(d) Quick sort

Part (i) [1.5 MARK] For which one of these cases is it necessary to apply a sorting algorithm to arrange elements in ascending order?

(a) In-order traversal of a binary search tree

(b) In-order traversal of an AVL tree

(c) Removing elements from a priority queue implemented via a min-heap

(d) In-order traversal of a binary tree

Part (j) [1.5 MARK] The complexity of operations on a hash table such as insertion and deletion is

(a) $O(1)$ if the table is half full

(b) $O(1)$ all the time

(c) $O(n)$ is the table is half full

(d) $O(log(n))$ all the time

Part (k) [1.5 MARK] Which one of the following is true of collision resolution policies of hash tables?

(a) Linear probing is the superior technique as the search for the next available location does not involve too many arithmetic computations

(b) Double hashing performs better as all keys have the same probing sequence

(c) The performance of separate chaining is not as good as linear probing as colliding elements are stored into a linked list

(d) Double hashing performs better as each distinct key has different probing sequences

Part (l) [1.5 MARK] Which one of the following does not define a graph?

(a) A graph is completely defined by enumerating all paths possible between all nodes in the graph

(b) A graph is defined by taking the union of adjacency sets of all nodes in the graph

(c) A graph is completely defined by including all possible paths from a particular node in the graph

(d) A graph is completely defined by sets V and E that contain all nodes and edges in the graph

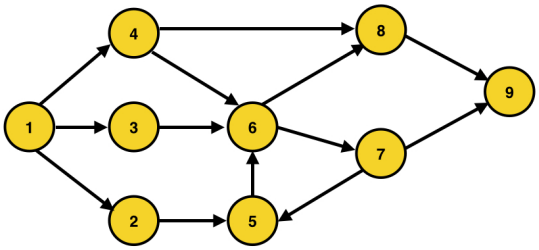

Figure 1: Graph 1

Part (m) [1.5 MARK] Breadth-first traversal of the graph in Figure l starting from node 1 results in the sequence:

(a) 1-2-3-4-5-6-8-7-9

(b) 1-4-3-2-6-5-8-7-9

(c) 1-2-5-6-7-9-3-4-8

(d) 1-2-5-6-7-9-8-3-4

(e) None of the above

Part (n) [1.5 MARK] Depth-first traversal of the graph in Figure l starting from node 1 results in the sequence:

(a) 1-2-3-4-5-6-8-7-9

(b) 1-4-3-2-6-5-8-7-9

(c) 1-2-5-6-7-9-8-3-4

(d) 1-2-5-6-7-9-3-4-8

(e) None of the above

Part (o) [1.5 MARK] Which one of the following statements is correct about the graph in Figure l?

(a) Predecessors of 6 are 5, 3 and 4; Successors of 6 are 7 and 8; In-degree of 6 is 3; out-degree of 6 is 2

(b) The graph is strongly connected

(c) Predecessors of 6 are 7 and 8; Successors of 6 are 5, 3 and 4; In-degree of 6 is 2; out-degree of 6 is 3

(d) The graph is unconnected

Part (p) [1.5 MARK] What is the topological ordering of nodes in the graph in Figure l?

(a) 1-4-3-2-6-5-8-7-9

(b) 1-2-5-6-7-9-8-3-4

(c) 1-2-3-4-5-8-6-9-7

(d) 1-2-5-6-7-9-3-4-8

(e) None of the above

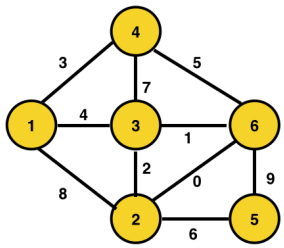

Figure 2: Graph 2

Part (q) [1.5 MARK] The minimal spanning tree of the graph in Figure 2 is composed of the edges:

(a) (1-4),(1-3),(3-6),(2-6),(2-5)

(b) (5-2),(2-6),(6-3),(1-4)

(c) (1-2), (2-5), (5-6), (6-4), (4-3)

(d) (2-4),(1-3),(5-6),(2-6),(2-5)

(e) None of the above

Part (r) [1.5 MARK] The Prim's algorithm used to compute the minimal spanning tree:

(a) Uses a non-greedy method to find the edges forming the minimal spanning tree in the graph

(b) Uses a greedy algorithm that is locally and globally optimal

(c) The Prim's algorithm can be used to find the shortest path between two nodes in the graph

Part (s) [1.5 MARK] At every iteration of the shortest path algorithm, a node is selected to be added to the set W if:

(a) The weight on the edge connecting the selected node to the previously added node is the least

(b) It has the least distance to nodes in the set W

(c) It has the least number of edges connecting it to the starting vertex

Part (t) [1.5 MARK] Which one of the following suffices to conclude that a problem is NP-complete?

(a) The solution of the problem can be verified in polynomial time and is reducible to another NP-complete problem in polynomial time

(b) If the problem can be shown to be NP-hard

(c) If the problem can be shown to be NP

(d) If the problem can be solved in exponential time by a non-deterministic machine

Question 2. [15 MARKS]

Implement the following interface functions of a queue and define the structures `Queue` and `Node` appropriately. The `Node` structure should consist of a member to store an integer data value. The underlying implementation of the queue must be linked (i.e. not array). Your implementation must not result in memory leaks.

- `struct Queue * initQueue();`
  - This function dynamically allocates memory space and initializes a `Queue` structure.
- `int dequeue(struct Queue * q);`
  - This function removes a `Node` from the queue `q` and returns the `integer` data member stored in this node.
- `void enqueue(struct Queue * q, int data);`
  - This function creates a node and assigns the `data` value into one of the member fields.
  - Then it inserts this node into the queue `q`.

.

.

Question 3. [15 MARKS]

Suppose that you are provided with a pointer to a linked list. Each node in the linked list has the following structure:

```
struct Node{
   int data;
   struct Node * next;
}
```

You are to sort nodes in this linked list using bubble sort. You should manipulate pointers in existing nodes of the linked list so that these are rearranged in ascending order of `data` elements. Your implementation must be in-place (i.e. you cannot just exchange `data` members in each node, you cannot use any arrays in your implementation, etc). Your implementation can be recursive or iterative. The function you will define has the following prototype:

- `void bubbleSort(struct Node ** l1);`
  - `l1` is the address of the pointer to the first node in the linked list
  - This function has no return values

You can define additional helper functions if needed. Comment on the pros and cons of sorting a linked list rather than an array.

.

.

Question 4. [15 MARKS]

Consider a directed graph containing `N` nodes represented by an adjacency matrix `aM`. In this question, you will implement a function that verifies whether a node `m` is weakly connected to all other nodes in the graph. Each node in the graph is labelled from 0 to `N-1`. The adjacency matrix of a graph containing `N` nodes is defined by the structure `adjMatrix`:

```
struct adjMatrix{
   int aM[N][N];
};
```

If there is an edge from node i to j, `aM[i][j]` is 1 otherwise `aM[i][j]` is 0. Assume that the following function definitions are available to you:

- `struct Queue * initQueue();`
  - This function dynamically allocates memory space and initializes a `Queue` structure.
- `int dequeue(struct Queue * q);`
  - This function removes a node from the queue `q` and returns the `integer` data member stored in this node.
- `void enqueue(struct Queue * q, int data);`
  - This function creates a node and assigns the `data` value into one of the member fields.
  - Then it inserts this node into the queue `q`.
- `int isEmpty(struct Queue * q);`
  - This function returns 1 if the queue `q` has no nodes and returns 0 otherwise

Implement the following function which returns 1 if node `m` is weakly connected in the graph represented by `aM` and 0 otherwise.

- `int isConnected(struct adjMatrix * aM, int m);`

You can define helper functions if needed.

.

.

Question 5. [25 MARKS]

Unlike AVL trees, operations such as insertion into a binary search tree do not preserve the structure of the tree. It is possible to have an extremely skewed binary search tree after several insertions. In this question, given the pointer to the root of a binary search tree, implement a recursive function that balances the binary search tree (hint: some elements of this implementation are similar to inserting into an AVL tree). A node in the binary search tree is defined according to:

```
struct Node{
  int height;
  int data;
  struct Node * lChild;
  struct Node * rChild;
};
```

Assume that you are provided with the following helper functions (you must use these in your implementation):

- `struct Node * rightRotate(struct Node * n);`
  - This function will perform a single right rotation around the root `n` of a subtree and return the new root pointer of the rotated tree
- `struct Node * leftRotate(struct Node * n);`
  - This function will perform a single left rotation around the root `n` of a subtree and return the new root pointer of the rotated tree
- `int height(struct Node *n);`
  - This function returns value in the `height` member of `n` if `n` is not `NULL` and 0 otherwise
- `int max(int i, int j);`
  - This function returns the maximum of two integers `i` and `j`

Implement the following function:

- `struct Node * balanceBST(struct Node * n);`
  - This function returns the root pointer of the balanced binary search tree whose original root is `n`

.

.

[Use the space on this page for rough work. Indicate clearly any work you want us to mark.]

[Use the space on this page for rough work. Indicate clearly any work you want us to mark.]

[Use the space on this page for rough work. Indicate clearly any work you want us to mark.]

Aid Sheet

- **Arithmetic sequence**:

$$S = \sum_{i=0}^{n-1}(a + id) = \frac{n}{2}(2a + (n-1)d)$$

- **Geometric sequence**:

$$S = \sum_{i=0}^{n-1} ar^i = \frac{a(1-r^n)}{1-r}$$

- **Heaviside cover-up** rules for partial fraction expansion:

$$\frac{ax^2+bx+c}{(x-d)(x-e)^2} = \frac{A}{x-d} + \frac{B}{x-e} + \frac{C}{(x-e)^2}$$

$$A = \frac{ad^2+bd+c}{(d-e)^2} \quad C = \frac{ae^2+be+c}{(e-d)}$$

B : **solve for B** $\quad \frac{ac^2+bc+c}{(c-d)(c-e)^2} = \frac{A}{c-d} + \frac{B}{c-e} + \frac{C}{(c-e)^2}$

where c is a constant