UNIVERSITY OF SASKATCHEWAN
DEPARTMENT OF COMPUTER SCIENCE
CMPT 250
Final Examination
3 hours Closed Book April 22, 2002
Marks

This exam is out of 150. Thus, you have 1 minute per mark with 30 minutes to spare. Use this in judging how much time to spend on a question.

---

1. 1. Each data structure has its advantages and disadvantages. Selecting the appropriate data structures under different situations is important to the performance of the system. Please (i) specify the ideal data structure that you should use in the following situations, and (ii) give the running time (in big-O notation) of it.

   5 (a) SaskTel provides an online directory service of all current Saskatoon subscribers that can be searched by subscriber's last name to obtain the personís phone number and address. If the last name input does not exist, the people that have names close to the given name in alphabetic order are also listed. Assume all data can be held in memory, and most Saskatoon citizens use this service frequently to search for their friendsí phone numbers.
   Specify the data structure to be used, and the time required for a search.

   5 (b) SaskTel provides a directory service to video stores in the city so that the video stores can find the street address from a phone number when opening accounts for new customers. Assume all data can be held in memory.
   Specify the data structure to be used, and the time required for a search.

   5 (c) U of S provides a directory service for the reference books used by U of S classes during the year. Students search for books by providing a book title. If the book title input does not exist, books that have titles close to the given title in alphabetic order are listed. The reference books used by the classes are different for every assignment and most students only search for the reference books for the current assignment. Assume all the data can be held in memory.
   Specify the data structure to be used, and the time required for a search.

6 2. How can polymorphism be used to build a better system? In answering this question, you might consider the polymorphism design pattern that was discussed after the banking case study.

5 3. Give a collection of test situations for the insertion of an edge into a graph where the edges are stored in adjacency lists.

22 4. In Eiffel, define a descendant of the class LINKED_SIMPLE_TREE_UOS[G] that stores STRINGs and has one added feature, a BOOLEAN function called *value*. The function is to return the BOOLEAN value associated with the current tree. In each leaf, the STRING will be either the string "true" or the string "false", representing the corresponding BOOLEAN value. In an interior node, the STRING will be either "and", "or" or "not", representing the corresponding BOOLEAN operation that is to be applied to the subtree values. If an interior node has the string "not", then the left subtree stores a BOOLEAN tree and the right subtree is Void. You can assume that the current tree has

valid values in its leaves and interior nodes. The function *value* should be designed to only handle non-empty trees. As an example, the following tree has value *False*.

For reference, the main public features of LINKED_SIMPLE_TREE_UOS[G] are

- make
- is_empty : BOOLEAN
- initialize (lt : like Current; r : G; rt : like Current)
- root_item : G
- root_left_subtree : like Current
- root_right_subtree : like Current
- out : STRING
-

9 5. When handling records stored in a binary file, several records are stored in a block and a file operation consists of reading or writing a complete block. For **each** of the following file organizations, what advantages, if any, are there to having many records in a block:

a. sequential file,
b. direct file implemented by using a hash function,
c. B-tree file that can handle sequential operations and direct access.

15 6. In the U-Star system for one session, both the students and classes of U of S need to be managed. Assume only the data for one session needs to be managed, all data can be stored in memory, and the following operations need to be handled:

a. search for the class information given the class call number (e.g., the call number for CMPT 250 section 01 is 86525001),
b. search for the student information given the student number,
c. after a student record is found, register/drop a class in this session for this student,
d. add a new class into the system,
e. after a student record is found, list all classes in which this student is registered,
f. after a class record is found, list all students registered in this class.

Keep in mind the following space constraints:

- There should be only one copy of each studentís information (e.g., name, home address, phone number, email address). There can be multiple copies of the student number.
- There should be only one copy of each classís information (e.g., abstract, credit units, instructor, textbook). There can be multiple copies of the class call number (e.g., "85625001").

Please design an appropriate data structure and algorithms that can handle the above operations and meet the space requirements. You can use English description, diagrams, pseudo code, Eiffel syntax, or any combination of these to explain your design. Keep your explanation as brief as possible. Include the time requirements for each of the above operations.

7. Recall that in a weight-balanced tree, every key value has an activity count that is stored with it in a node. Also, a weight-balanced tree is an ordered binary tree ordered by the key values arranged so that the key with the largest activity count is at the root. In the diagram of a weight-balanced tree below, in each node to the left of the ":í is a letter that is the key for the node. To the right of the ":" is an integer that is the activity count for the node.

3 (a) For the weight-balanced tree below, draw a diagram of the tree after an access of the key a.
4 (b) For the weight-balanced tree above (**before** the access of part (a)), draw a diagram of the tree after an access of the key e.
3 (c) For the weight-balanced tree above (**before** the access of part (a) or part (b)), draw a diagram of the tree after the deletion of the node with key g.

12 8. Consider the following algorithm that does a depth-first search of an undirected graph G=(V, E) from vertex v_0. Note that it inserts into container T the edges to unreached vertices. At the end, T has the edges that form the depth-first search tree from v_0.

```
Main program
    For each vertex u
        u.set_reached(false)
    end
    T.make
    scan(v0)
end
scan (s) algorithm
    s.set_reached(true)        <- s.set_num_descd(0)
    For each w adjacent from s do
        if not w.reached
        then
                T.insert((s,w))
                scan(w)
                s.set_num_descd(w.num_descd+1 + s.num_descd)
        end
    end
end
```

For the following graph, the depth-first search tree of vertices connected to vertex 1 is given: 1

1 : 6 9 7
2 : 8 6 5 9 6 7
3 : 7
4 : 6
5 : 2 6 2 4 3
6 : 2 5 4 1 8
7 : 1 3
8 : 2 6 8 5 9
9 : 2 1

Add appropriate statements to the main program and/or scan algorithm to do each of the following tasks. Note that there are simple efficient ways of doing each of them. Inefficient and/or overly complex solutions will only obtain partial marks. Both tasks can be handled together, or they can be handled separately.

a. For each vertex reached during the depth-first search, except the start vertex v_0, determine its parent with respect to the tree. For this assume that each vertex has a procedure called *set_parent*(p : VERTEX) to assign its parent, and a function *parent* : VERTEX to retrieve its parent. For example, parent of 5 is 2.
b. For each vertex reached during the depth-first search, determine the number of vertices marked reached during the scan from it, i.e., the number of descendants that it has in the depth-first tree. Note: do not count the vertex itself in its own count. For this assume that each vertex has a procedure to set its value, called *set_num_descd*(i : INTEGER), and a function to access its value, num_*descd* : INTEGER. For example, vertex 6 has 5 descendants.

6 9. Suppose that into the scan algorithm of the previous question, two procedure calls proc1 and proc2 have been added, as shown below. Suppose that the graph is undirected and connected with n vertices and m edges. Also, assume that the time requirements for proc1 and proc2 are as follows:
$T_{proc1}(p) = O(p)$
$T_{proc2}(q) = O(q)$
for some parameters p and q. Given the order for the time requirements for the call scan(v_0) in the main program. Do not assume that there is any relationship between either p or q and either n or m.

```
scan (s) algorithm
    s.set_reached(true)
    For each w adjacent from s do
        if not w.reached
        then
            T.insert((s,w))
            scan(w)
            proc1
        end
        proc2
    end
end
```

10. **A Point-of-Sale Application**
This case study involves modeling a point-of-sale system for a grocery store, such as Safeway or Superstore. The purpose of the system is:

> "To help each cashier work more effectively during customer checkout, to keep records of each customer sale, to maintain inventory of store items, and to support more efficient store operations."

The system is to support the following:

1. The logging of important information:
    - to maintain the price of each item that is to be accessed by its universal product code (UPC)
    - to maintain tax categories, including rate and effective date
    - to maintain the authorized cashiers (including head cashier) for the store

- to maintain what items are sold in the store ó reordering further items (based on reorder quantity) from suppliers when quantity on hand becomes too low (based on reorder level); add new items from new or existing suppliers; and discontinue certain items
- to log the result of each customer sale in the store

2. Conducting day-to-day business:
   - to subtotal the prices of the items for a customer sale, including any sales taxes, and obtain a total amount due for the customer sale
   - to accept payment by cash, check, or credit card

3. Facilitate the store manager analyzing business results:
   - to count how many of each item was sold
   - to count how much was received in cash, check, and credit card sales
   - to assess how each cashier is performing
   - to assess how the store is performing

4. Working with interacting systems:
   - to obtain authorization from one or more credit card or check authorization systems
   - to report and transfer sales taxes to the provincial and federal government systems

Customer business in the store is handled through point-of-sale terminals called registers. Each item is swiped/scanned for its UPC code and priced. A customer sale consists of various store items, where each item purchased has an associated quantity that specifies how many were purchased. The total of the customer sale, including the calculated tax, is produced. Any change due to a customer on a cash sale is to be calculated. A detailed record of a customer sale is to be printed and given as a receipt to the customer. Each item has an associated description: UPC code, price, tax category, and quantity on hand. A store has many things, such as cashiers, registers, and items for sale. A sale session involves a cashier and a register. A cashier starts a session on a specific register on a specific date and time. Sessions are used to assess (or evaluate) cashier performance. To evaluate a casherís performance, you need to know all the corresponding sessions for that cashier. Many customer sales will be associated with a session. Also, a customer sale has an associated payment.

The deliverables for this question consist of the following:

8 (a) A list of actors and use cases for the application.

20 (b) Two interaction diagrams (either sequence diagrams or collaboration diagrams), one for each of

- a customer sale consisting of the purchase of several different items;
- assessing a specific cashier for performance.

15 (c) An overall (complete or composite) class diagram for your system, i.e., include all classes and their associated relationships for your system.

7 (d) Any inheritance diagrams with important attributes and routines.

Total 150 The end