Spring 2002 Dr. Ching

CALIFORNIA STATE UNIVERSITY, SACRAMENTO
College of Business Administration

MIS 211 - Information Systems II

Midterm Exam

Points: 100 Date Due: Wednesday, March 20

Answer the following three problems. The problems have been written for yo to reflect upon our discussions and readings. Be sure to form your solutions and answers in the context of the textbook, article reading, and class lectures and discussions. Please keep in mind that credit cannot be awarded for assumed work.

Note. Your answers should reflect your individual effort. Collaborating with others, regardless of whether they are enrolled in this class or not, will lead to a failing semester grade. The uniqueness of everyone's logic and the incorporation of the OO programming concepts that have been presented differentiate programming solutions to the extent that collaboration can be easily seen. Answers showing close resemblance will be considered a collaborative effort.

Guidelines

1. The exam should reflect an individual's effort, and NOT be the product of group collaboration. Any form of copying, sharing, assisting, seeking aid from another person, or submitting work that has not been completed by you (either in part or entirity) will lead to an automatic failure for the semester.

2. All answers should be software generated. A penalty of 15 percent (of the total possible points for the question) will be assessed.

3. Please do NOT bind your exam answers. Staple each answer in the upper left corner and submit them in a 9 × 12-inch envelope.

Refer to the syllabus for other requirements.

Recyclable paper. Please recycle.
Reduce, Reuse, Recycle

Part 1 - Data Management

1. B+ trees (35 points)
   a. Using the C++ program (main.ccp) contained in exam1.exe, generate a list of keys and diagram them in a B++ tree (the list will be contained in the *prob1.txt* file).

   Note. Main.ccp will generate a list of 12 random numbers. Be aware that every time you run the program, a new list of numbers will be generated for <u>both</u> prob1.txt and prob3.dat (see problem 3).

   Specifications:
   - Order = 3
   - Depth = 3
   - The leaf nodes are initially evenly distributed among the nodes.

   Note. Do NOT create the tree by entering the keys individually (i.e., one by one). Use all the keys to represent the initial state of the tree.

   Please note the corrections to Figure B.11 on page 995 of your textbook. The first (leftmost) pointer points to the lowest value of the node's child or descendant in the case of the root node. The pointer corresponding to the <u>first key</u> points to the starting (lowest) value of the node to the right of the first node. In the case of the root node, this would be the second branch. The pattern is repeated for subsequent points in accordance to the order and other rules. Also, note the order cannot be equal to 1.

   b. Insert the five keys listed under *Insert Keys* (in their listed order) into the tree.

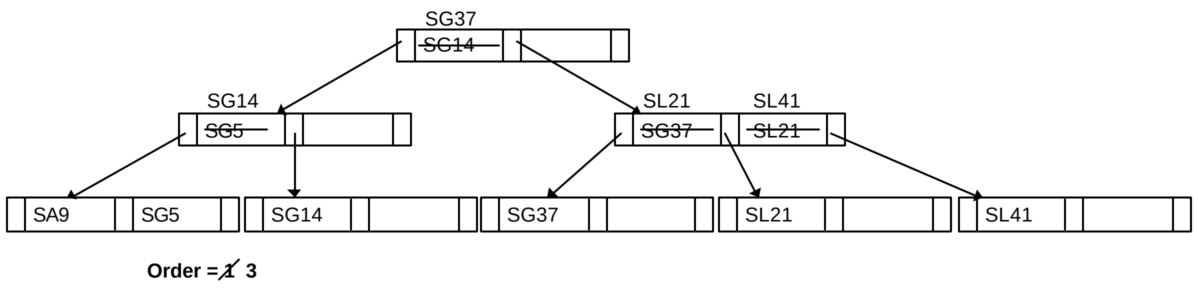


Figure 1. Corrections to Figure B.11 on page 995.

<u>Tangibles</u>

- Submit diagrams of the initial tree and the tree after each of the five keys has been inserted. Your diagram should be software-generated (you should have a total of 6 diagrams).

- Include the list of keys generated by the C++ program with your diagrams. Otherwise, your answer cannot be graded.

Part II - OO Programming

2. Array Processing (30 points)

Create an OO C++ program that reads the sales.dat file, and accumulates the sales revenue and volume for the manufacturers of the various computer products, and produces a report that lists each manufacturer and their accumulated totals.

Create a class and instantiate two objects, sales revenue and sales volume, in them. The class will contain a single numeric array for the accumulations (before you can use the array, you must initialize all elements to zero). Methods must be used to access all of the object's variables and data. A character string array for the manufacturer names can be declared in either the main function or class.

Inputs

There are two files (contained in exam1.exe), manuf.dat and sales.dat. Manuf.dat contains the manufacturer names that will appear in the report. Read its records into a character string array.

Manuf. dat

Field	**Data type**
Manufacturer name	char(25)

Sales.dat contains the products and their annual sales revenue and volume. A pointer field specifies the location (array element) into which they must be accumulated as well as the product's manufacturer location (element) in the manufacturer name array. The three locations will correspond to one another. For example, Xerox is the thirty-eighth manufacturer. Its location in the manufacturer names array is 37. Therefore, its sales revenue and volume (for all its products) must be accumulated in the thirty-seventh location (i.e., as specified by the array location field) in the sales revenue and volume (object's) array.

Sales. dat

Field	**Data type**
Stock number	char(16)
Array location	int or short
Sales revenue (in thousands)	double
Sales volume	double

Output

Sequentially access the arrays (beginning at the zero-th element) to produce your report. Design your report to include the following:

- Page and report heading. The page heading consists of a title and column headings. The report heading describes the contents of the report, and appears centered.
- Detail lines. Aesthetically arrange the output over 80 columns (8½ × 11-inch page, portrait orientation). Include the manufacturer's name, sales revenue and volume.
- Report footing. The report footing shows the column totals for sales revenue and volume (i.e., all manufacturers).
- Aesthetic design. Aesthetically arrange the above items in your report. These include (but are not limited to) column alignment, report and column title placement, and vertical and horizontal spacing (i.e., spaces between and within the line).

Tangibles

- Hardcopies of both the output (report.txt) and program (i.e., source listing).
- A 3½-inch diskette or any other medium with your C++ workspace. If necessary, you may compress everything (in the workspace) into a zip file.

3. Bubble Sort (35 points)

Modify your last programming assignment (homework 3/4) to perform a bubble sort. Follow the flowchart on the following page to program the sort (because this is a generic algorithm, you will have to make adjustments for C++). Printout the results of your sort (do not generate your search report).

Note. Main.ccp will generate a list of random numbers. The number of records will vary according to a random number between 50 and 100. Be aware that every time you run the program, a new list of numbers will be generated for both prob1.txt and prob3.dat.

Design/adjust your program to meet the following specifications:

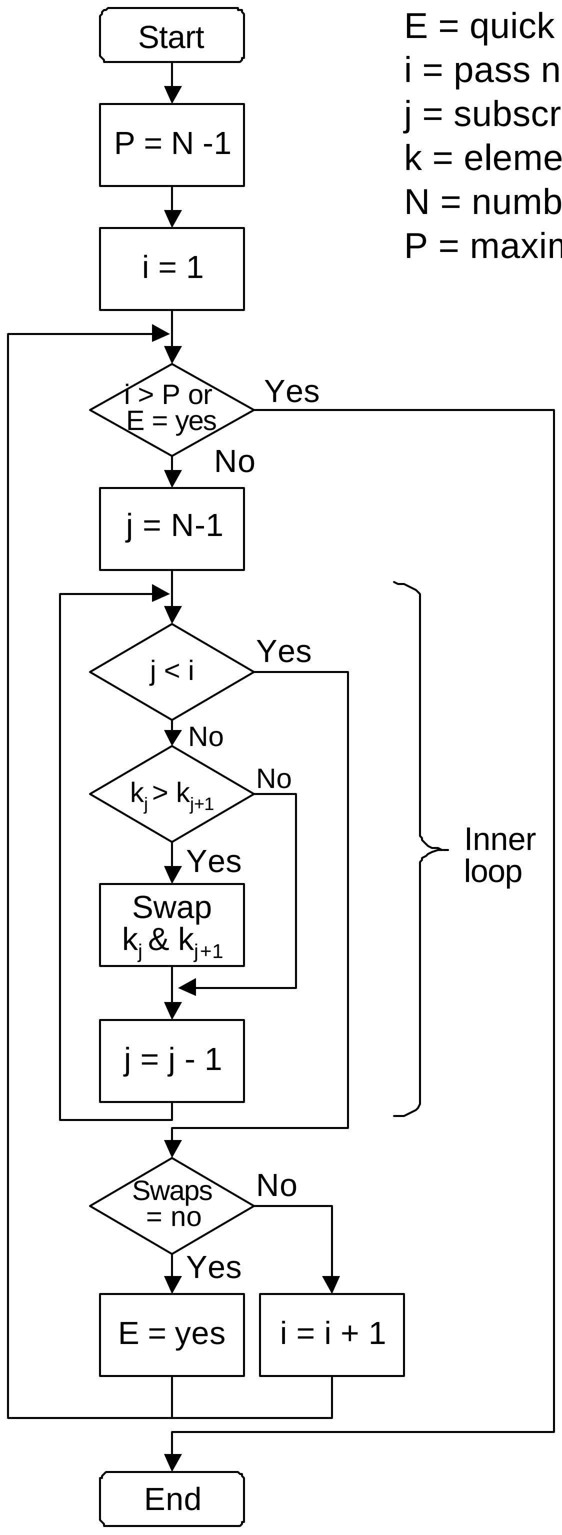
- Create a third subclass for the bubble sort.
- Create a method in the (third) subclass for the bubble sort.
- Instantiate an object in the (third) subclass.
- The object's data can only be accessed through its methods:
    - Load data from the prob3.dat file (generated by exam1.ccp) into the object's array. You will need to modify your *open* command. Be sure your arrays have been declared for 100 elements. You will need to count the number of records read into the arrays (this is N in the flowchart).
    - Bubble sort the numbers in the stock number array.
    - Print the contents of the sorted array along with the number passes (i.e., number of times the outer loop executed) and the total number of compares (for all times the inner loop was executed).

Tangibles

- Hardcopies of both the output (report.txt), prob3.dat file and program (i.e., source listing).

  Note. Be sure to include a printout of prob3.dat. Otherwise, your answer cannot be graded.

- A 3½-inch diskette or any other medium with your C++ workspace. If necessary, you may compress everything (in the workspace) into a zip file.



E = quick exit
i = pass number (values 1 through P)
j = subscript (values N-1 through i)
k = element value
N = number of elements in the array
P = maximum number of passes

Swap means to exchange the locations (what is in location j goes into j+1, and vice versa)