EHB354E - FINAL EXAM
10/6/2023

- Use the Microsoft Word file (Answers File) that is provided on Ninova, to write your answers.
- When you finish answerings, save the Word file on your computer and exit from Word program.
- Submit the Word file to Ninova from Homeworks section.

- Student should answer the questions without getting any help or collaboration.
- Automatic plagiarism detection software will be used for scanning the submitted answers.
- The answers that have significant similarities between them will be graded as zero.

QUESTION 1) [20 points] Write the complete C++ program to do followings.

Define a class named **Subcategory**, with public members below:

Name (string), Amount (integer), Parametered constructor, Print function.

Define a **STL map** (M) object (Standard Template Library).
The Key field of M map is a Group name string, and the Value field of M map is a **STL vector** of Subcategory class.
The following dataset contains group names, subcategory names, and subcategory amounts.
By using the given dataset, add group names to M map as Keys , and vectors of Subcategory class to M map as Values.

ELECTRONICS GROUP	
Router	41
Battery	136
Monitor	30
Printer	24
Audio	43
Navigation	15
Video	32
Modem	26

MACHINERY GROUP	
Shipping	23
Crane	7
Hydraulics	15
Pneumatics	10
Forklift	8
Carrier	13

STATIONERY GROUP	
Paper Clip	1370
Envelope	560
Ink	140
Notepad	710
Paper Pack	230
Clipboard	1720
Binder	690

By looping thru the groups in M map, perform the followings.

Call the built-in **STL sort** function to **sort** the subcategory vectors in each group, from smallest amount to biggest amount.
Calculate the total of amounts for each group.
Generate a text file named **"Output.html"**, so that it contains tags for html tables for each Group.
A shortened example of Output.html file which contains only one table (partial), and how it is shown in web browser is given below.
Contents of output file should be written programmatically, by looping and using the M map you have defined.
All html tables should be written to output file one after another, so that in web browser they are shown in vertical order.
Program will not take any user inputs from keyboard or from an input file, and it will not display any outputs on screen.

Example (partial) of Output.html File :

```
<table border=1>
<tr>
<td  align=center colspan=2 bgcolor=aqua > STATIONERY GROUP </td>
</tr>
<tr>
<td> SUBCATEGORY NAME </td>
<td> AMOUNT </td>
</tr>
<tr>
<td> Ink </td>
<td  align=center> 140 </td>
</tr>
<tr>
<td> Paper Pack </td>
<td  align=center> 230 </td>
</tr>
<tr>
<td> TOTAL AMOUNT </td>
<td  align=center> 370 </td>
</tr>
</table>
<br>
```

Example of html table shown in web browser :

STATIONERY GROUP	
SUBCATEGORY NAME	AMOUNT
Ink	140
Paper Pack	230
TOTAL AMOUNT	370

QUESTION 2) [10 points] Draw UML Sequence Diagram and Communication Diagram for the C++ program below.

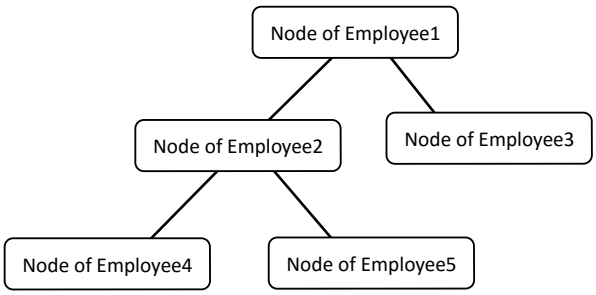
```
class C {
  public:
  void f3() { f4(); }
  void f4() {}
};

class B {
  C *  c;
  public:
  B() : c (new C) {}
  void f2(bool x) {
     if (x)  c->f3();
  }
};
```

```
class A {
  B  b[5];
  public:
  void f1() {
     for (B  j  :  b)
         j.f2(true);
  }
};

int main()
{
  A  a;
  a.f1();
}
```

QUESTION 3) [70 points] The following is an example of **Binary Search Tree** (BST) which is an ordered data structure containing nodes.
Each node is composed of three fields: Employee struct, Left branch pointer, Right branch pointer.
Trees will be maintained in alphabetical order by the Employee names.
Alphabetical Order Rule : In terms of Employee names in nodes, all nodes in left subtree are smaller than root node (top node), and all nodes in right subtree are bigger than root node.



Write all C++ codes for member functions of BST class given below. (Do not make any modifications to the class specifications.)

```
struct Employee {   // Employee struct definition
   int ID;          // Employee ID number (unique)
   string Name;     // Employee surname (not-unique)
};


// Binary Search Tree
class BST
{
public:
   struct Node {          // Nested struct definition
     Employee  emp;       // Data
     Node *left, *right;  // Pointers to left and right branches
   };
  string deptname;                          // Department name
  Node * root;                              // Pointer to top node
  BST (string deptname, Employee kaynak[] , int N); // Constructor1
  bool ADD (Employee  yeni);                // Iterative
  BST (int N);                              // Constructor2
  BST (const  BST &  other_tree);           // Copy constructor
  void DUPLICATE (Node * otherP);           // Recursive
  Employee * SEARCH (string isim);          // Iterative
  Employee * SEARCH (int idnum, Node * P);  // Recursive (overloaded)
  void DISPLAY (Node * P);                  // Recursive
}; // end of class
```

[5 points] CONSTRUCTOR1 : Takes department name, array of employees, and size of array as parameters.
By looping thru elements of array parameter, calls ADD function, in which new nodes are added to tree.

[10 points] ADD Function (Iterative) : Takes a new Employee struct as parameter. By looping from root to leaf nodes (bottom nodes), finds a suitable leaf node in tree, and adds a new allocated node to either left branch or right branch of leaf node.
Function should check whether the same employee ID already exists in tree. If so, function returns false without adding a new node.

[3 points] COPY CONSTRUCTOR : Takes another tree as parameter, and calls `DUPLICATE` function.

[5 points] DUPLICATE Function (Recursive) : Takes root pointer of another tree as parameter. Recursively travels all branches in the other tree and calls ADD function, so that all data are duplicated from other tree to the tree itself.

[7 points] CONSTRUCTOR2 : Takes only the size (N: number of nodes in tree) as parameter.
By looping, adds N nodes to tree, each node containing randomly generated data (random employee ID integer and random employee name string). Department name of tree is assigned as "Testing".

[8 points] SEARCH Function (Iterative) : Takes an Employee name as the search parameter. By looping, searches the tree from root to a leaf node. Function should convert names to all uppercases when doing string search. Partial string matches should be accepted as valid.
If the searched Employee name is found (exact match or partial match), then function returns a pointer to Employee struct found, otherwise returns NULL.

[7 points] SEARCH Function (Recursive, Overloaded) : Takes an Employee ID number as the search parameter, and the root pointer as second parameter. Recursively travels all branches in tree. If the searched Employee ID number is found (exact match required), then function returns a pointer to Employee struct found, otherwise returns NULL.

[5 points] DISPLAY Function (Recursive) : Takes the root pointer as parameter.
Recursively travels all branches in tree and prints employee information of all nodes in tree.

[20 points] MAIN Program : By using member functions and constructors of BST class, perform the following tasks.

Suppose there are four departments in a company, each department has several employees.
For each department, information of employees will be stored in separate BST trees. All trees will be stored in one STL list.

Define the **STL list** (Standard Template Library) which will contain **BST trees.**
Add the following four trees to the STL list. (Do not use any other datasets.)
Use department names, employee ID numbers and employee names as constructor parameters.

Distribution	Accounting	Production	Marketing
101 - Sunay	201 - Şenol	301 - Çakır	401 - İlker
102 - Cihan	202 - Aktaş	302 - Sunay	402 - Vedat
103 - Üner	203 - Kaplan	303 - Acar	403 - Ceyhan
104 - Kaptan	204 - Sungur		

Also add the following two trees to the STL list.
Duplicated tree : Copy-constructed from tree of the "Distribution" department.
Randomly generated tree : Tree with size of 6 nodes. (Constructor2 will assign random data.)

By looping thru all elements in the STL list, call the DISPLAY function of each tree.

In an endless loop :
Display the following prompt message "Enter a search value (Employee ID integer or Employee Name string) : ".
Read user input from keyboard. (User may press Ctrl+C keys to exit from program.)
By looping thru all elements in the STL list, perform searching in all trees (Call the appropriate version of SEARCH function, based on the input entered by user.)
Display messages on screen about search results.