Middle East Technical University | Department of Computer Engineering

CENG 213

Data Structures

Fall '2014-2015

Programming Homework 2

Due date: 21 December 2014, Sunday, 23:55

1 Objectives

In this assignment, you are going to practice on stack and linked binary tree structure using C++. Your task is to implement a program to solve the linked 'and-xor' binary tree problems. You will use stack structure while traversing the tree.

Keywords: *Stack, Linked And-XOr Binary Tree, C++*

2 Problem Definition

Mr. Chef is working in a developing hotel in the city. He is trying to do best in cooking with quality ingredients. On the other hand, they have limited funds and have to consider the economical recipes in their cooking. Mr. Chef is good at cooking but is bad in arithmetic. Can you help him to calculate the economical recipes?

The recipes are represented as linked 'and-xor' binary tree. In this tree, root is the given problem. Each of the nodes in the tree can either be an 'and' node, a 'xor' node or a 'leaf' node. 'and' node requires solutions from both of its children, whereas 'xor' node requires a solution from exactly one of its children. 'leaf' node represents the ingredients of the recipe. Your task is to find the recipes along with their cost for the given food. There are some specifications about the problem which are given below:

- 'and' nodes and 'xor' nodes are the internal nodes which have exactly two children. 'leaf' nodes are the external nodes which do not have any children. Root node is also an internal node as well.
- The ingredients are represented on the 'leaf' nodes. Each 'leaf' node has also a cost value for the ingredient. Internal nodes do not have a cost value. You may assume it to be 0.
- 'leaf' nodes have also <name> data member for the ingredients. You may assume name value to be empty ("") for the internal nodes except for the root. The value of the <name> data member of the root is the name of the food.

3 Tasks

Your task is to find recipes along with their cost for the given food. To implement the solution, you have to complete the following implementations as well:

- Construct a stack class by implementing the interface given in Stack.h. Note that you are not allowed to make any changes in the interface.
- Construct a linked 'and-xor' binary tree class by implementing the interface given in LinkedAndXOrBinaryTree.h. Note that you are not allowed to make any changes in the interface. The methods and data members are explained below:
  - **struct Node:** This struct is used for the data information of a node in the tree.
  - **class Position:** This class is used for the manipulation of the position of a node in the tree. This is given to you. Just use it, you do not need to modify it.
  - **LinkedAndXOrBinaryTree():** Default constructor. Create an empty tree.
  - **void addRoot(NodeType _type,string _name):** Add root to the empty tree. <_name> is the parameter for the name of the food. Set cost data member as zero.
  - **void expandExternal(const Position& _p,NodeType _leftType,int _leftCost,string _leftName,NodeType _rightType,int _rightCost,string _rightName):** If the tree is not empty, add two new children to the internal node (Including root node). If the children are a 'leaf' (External) node, they have also a name (Ingredient name) and a cost (Ingredient cost). For the internal nodes, take name and cost parameters as "" and 0, respectively.
  - **bool empty() const:** Return true if the tree is empty.
  - **Position getRoot() const:** Return the position of the root. If tree is empty, return NULL position.
  - **Node* root:** This is the pointer data member to the root.
  - **class LinkedAndXOrBinaryTreeException:** This is an exception class. Throw this class when an invalid 'and-xor' tree operation occurs.
- Construct a recipe class by implementing the interface given in Recipe.h. Note that you are not allowed to make any changes in the interface. The methods and data members are explained below:
  - **struct Ingredient:** This struct is used for the data information of the ingredients of the recipe.
  - **Recipe(const BinaryTree& _tree):** Constructor for the recipe class. Input is the linked 'and-xor' binary tree of the food.
  - **void getArbitraryRecipe(vector<Ingredient>& arbitraryRecipe,int& totalCost):** Return one of the recipe of the food along with its total cost. The recipe is returned with input parameter arbitraryRecipe. The ingredients can be in any order, but there should not be missing ingredient or repeated ingredient. The total cost of the recipe is returned with input parameter totalCost.
  - **void getAllRecipes(vector< vector<Ingredient> >& allRecipes,vector<int>& totalCosts):** Return all the recipes of the food along with their total costs. Recipes can be in any order, but there should not be missing recipe or repeated recipe. The total cost of the recipes are returned with input parameter totalCosts. The index of a recipe in both allRecipes and totalCosts should be the same. The ingredients of a recipe can be in any order, but there should not be missing ingredient or repeated ingredient.
  - **const BinaryTree* tree:** This is the pointer data member to the linked 'and-xor' binary tree.

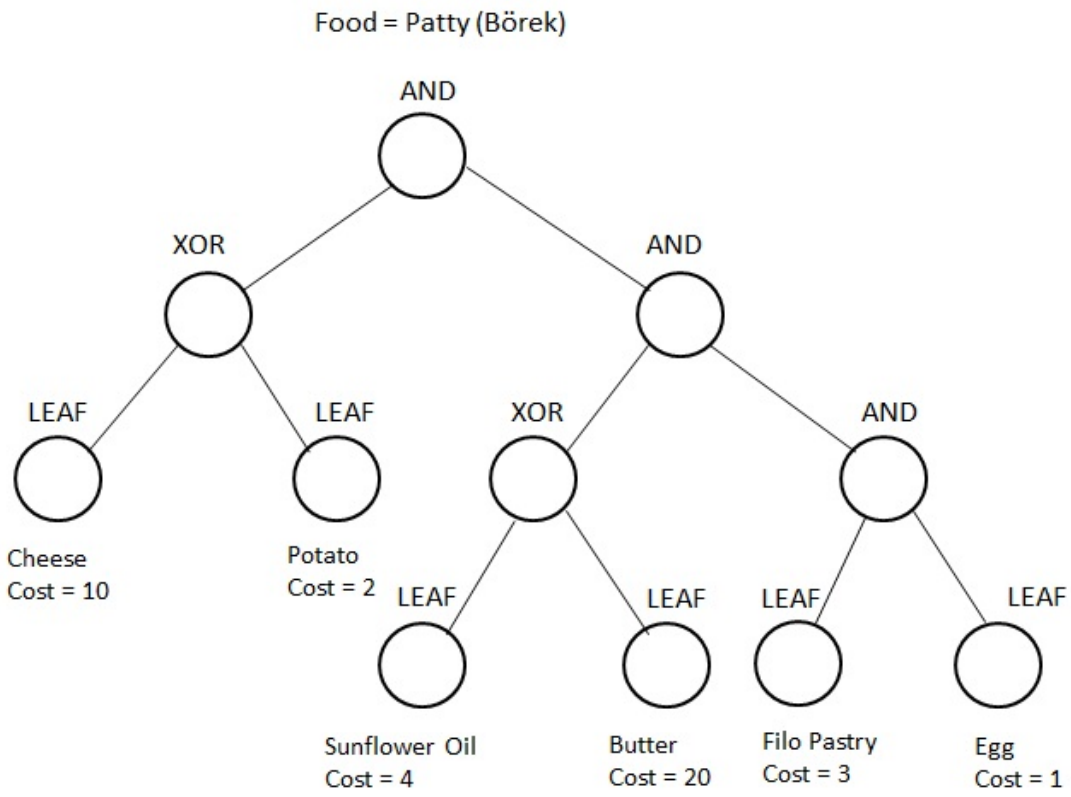

Figure 1: Linked And-XOr Binary Tree of the Food Patty.

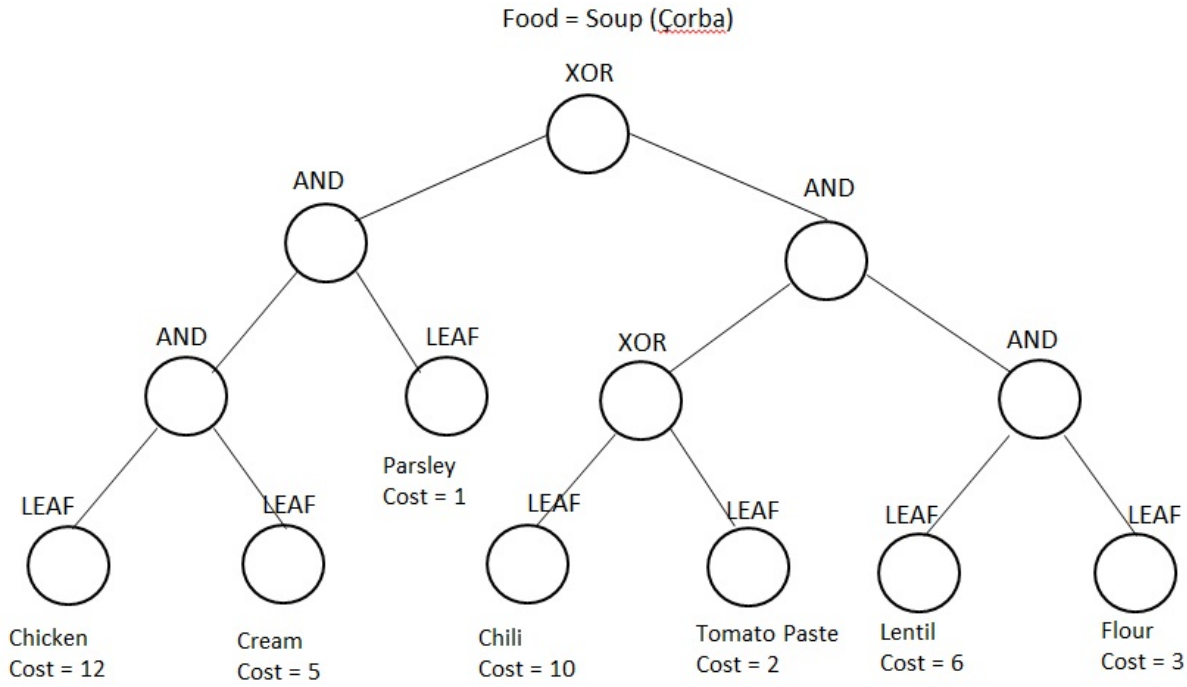

Figure 2: Linked And-XOr Binary Tree of the Food Soup.

4 Examples

- An example linked 'and-xor' binary tree for the food *Patty* is given in Figure 1. An example output of the food is as given below:
  The recipes of the Patty:
  Cheese(10), Butter(20), Filo Pastry(3), Egg(1) - Total Cost = 34
  Potato(2), Butter(20), Filo Pastry(3), Egg(1) - Total Cost = 26
  Cheese(10), Sunflower Oil(4), Filo Pastry(3), Egg(1) - Total Cost = 18
  Potato(2), Sunflower Oil(4), Filo Pastry(3), Egg(1) - Total Cost = 10

- An example linked 'and-xor' binary tree for the food *Soup* is given in Figure 2. An example output of the food is as given below:
  The recipes of the Soup:
  Chicken(12), Cream(5), Parsley(1) - Total Cost = 18
  Lentil(6), Flour(3), Chili(10) - Total Cost = 19
  Lentil(6), Flour(3), Tomato Paste(2) - Total Cost = 11

- There is an example driver program code which consists of linked 'and-xor' binary tree creation of these two examples, in the homework files.

5 Regulations

1. **Programming Language:** You should code your program in C++ language. Your submission will be compiled on department's inek machines. You are expected to make sure that your code compiles with g++ correctly.

2. **Late Submission:** You have a total of 7 days for late submission of the assignments. However, one can use at most 3 late days for any assignment. No penalty will be incurred for submitting within late 3 days. Your assignment will not be accepted if you submit more than 3 days late or you used up all 7 late days.

3. **Cheating: We have zero tolerance policy for cheating**. Sharing and copying any piece of code from each other and Internet is strictly forbidden. People involved in cheating will be punished according to the university regulations.

4. **Newsgroup:** You must follow the newsgroup (news.ceng.metu.edu.tr) for discussions and possible updates on a daily basis. Please ask your questions related to the homework on COW instead of sending email in order to your friends, who may face with same problem, can see your questions and answers.

5. **Evaluation:** Your program will be evaluated automatically using "black-box" technique so make sure to obey the specifications. 30% of the grade will be for stack implementations, 5% of the grade will be for linked 'and-xor' binary tree implementations and 65% will be for the results of different inputs. And also your code will be checked whether you traverse the tree with stack structure or not, so read the information about the homework carefully.

6. **Not Graded:** Below are some situations that your homework will not be graded (automatically assigned zero):

   (a) Traverse the tree without stack structure.

   (b) Solving the homework using any recursion.

   (c) Implementing the program by making any change in the interface files Stack.h, LinkedAndXOrBinaryTree.h and Recipe.h.

6 Submission

Submission will be done via COW. You will be given Stack.h, LinkedAndXOrBinaryTree.h and Recipe.h file. You will do Stack class implementation in the Stack.h file. Do the implementation below the given message in the Stack.h. You will do the implementations of the LinkedAndXOrBinaryTree class and Recipe class in the hw2.cpp file. You can also add some custom data members or functions to the hw2.cpp file. But make sure to obey the specifications of the homework !!

Create a tar.gz file named `hw2.tar.gz` that contains `Stack.h and hw2.cpp` files. Use sufficient comment lines in your algorithm in order to explain your solution clearly.

Note: The submitted archive should not contain any directories! The following command sequence is expected to run your program on a Linux system:

```
$ tar -xf hw2.tar.gz
$ g++ hw2.cpp main.cpp -o hw2
$ ./hw2
```

May it be easy.