|  |  |
| --- | --- |
| Name |  |

Cpt S 321 – Exam 2 – 40 points total

6 pages total (including one scratch page)

November 8, 2017

Notes:

* For the explanation problems, it is encouraged that you use diagrams if you need them as well as words to convey your ideas.
* For coding problems:
  + Full credit is given only for correct C# code (not pseudo-code or written explanations)
  + Unreadable / messy code is worth 0 points
  + Excessively length implementations and/or implementations that use strategies that are bad practice are worth few if any points
  + You are encouraged to create helper functions if you need them

1. (4 points) Implement the function below, *without* using LINQ, that takes a list of integers and returns a new list that contains only the even numbers from the original.

**List<int> CreateEvens(List<int> nums) {**

**List<int> evens = new List<int>(); // Use this in your code**

**foreach (int num in nums)**

**if ((num % 2) == 0**

**evens.add(num);**

**return evens; } // end of CreateEvens function**

|  |  |  |
| --- | --- | --- |
| 2. (2 points) What is the time and space complexity of your implementation for problem #1? (assume n items in nums list) | **Time** | **Space** |
| O(n) | O(n) |

3. (2 points) List 2 significant differences between a class and a struct in C#.

1. **Struct is value type so stored on stack; class is reference type so stored on the heap (and garbage collected)**
2. **Struct doesn’t support inheritance**
3. **Defaults: struct members are public, but class members are private**
4. **Array instances: for struct, allocated in-line (actual instances); class: reference type**

GRADER: more [here](https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/choosing-between-class-and-struct), read first.

4. (1 point) True or False? Every **class** and **struct** in C# inherits from object. **\_\_\_TRUE\_\_**

All struct inherit from System.ValueType, which in turn inherits from System.Object. A struct cannot inherit from anything else, however.

5. (3 points) Saving binary file formats usually requires less space than saving XML. So why is it that many modern software applications choose to write XML file formats as opposed to binary? List at least 2 different significant advantages that XML has over a binary format.

**Shivam: look for a few good test examples. Possibilities include: a lot of code to parse/analyze/emit XML code exists; disk space is cheap compared to programmer labor; etc. etc. etc.**

6. (3 points) For each of the following bits of code, assume they’re defined as a local variable in Main or some other valid function. For each one, write one of three responses for the result:

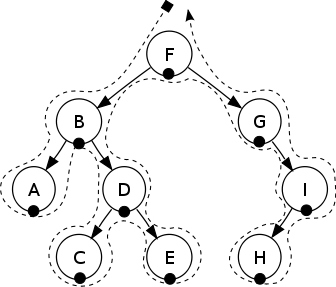
1. “does not compile” if the code will not compile
2. “runtime exception” if it will compile and run, but will produce an exception
3. the value of **s** after execution if the code compiles and runs without exception

|  |  |
| --- | --- |
| **Code** | **Result** |
| string s;  s += "Hello"; | Won’t compile |
| string s = null;  s += "Hello"; | Result: "Hello" |
| string s = "";  s += "Hello"; | Result: "Hello" |

7. (2 points) How are a frequency table and an anagram related, if they are? Explain, or state that they are not.

**Two words are an anagram iff (if, and only if) their frequency tables are identical.**

The next 3 problems use the following binary tree



8. (6 points) Give the order of traversal for each of the following

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pre-Order | **F** | **B** | **A** | **D** | **C** | **E** | **G** | **H** | **I** |
| In-order | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** |
| Post-order | **A** | **C** | **E** | **D** | **B** | **H** | **I** | **G** | **F** |

9. (2 points) Which order above is used to evaluate a binary expression tree?

**In-order.**

10. (3 points) Why are C# delegates a necessary feature, or at least extremely useful? (Hint: think of what you could NOT do, from what you have learned, without delegates.)

**You can’t have events without them. That is, how would you be able to register a callback without something like a C# delegate? (Or perhaps the cruder C++ function pointer, with a given method signature.)**

11. (1 points) Why does UML have not just Inheritance, but also Association, Aggregation, and Composition?

**Because there are more kinds of relationships between classes other than inheritance that can be usefully captured.**

12. (2 points) IF a node N in a binary tree has two children, what kinds of nodes can be children of N?

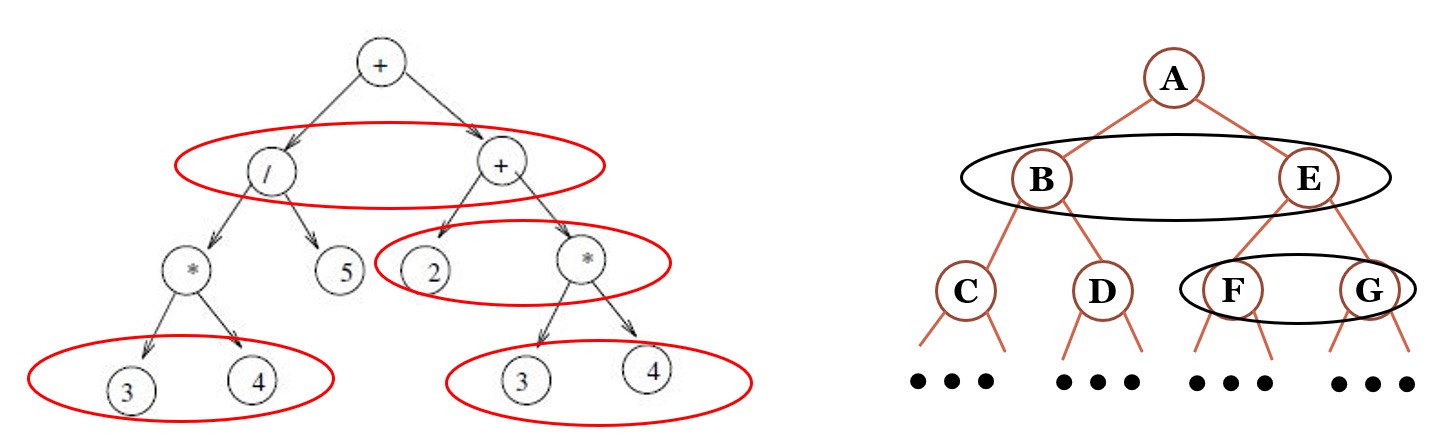
1. **Operand/Operator**
2. **Number**
3. **Variable**

13. (1 points) What is the relationship between then mathematical concept of commutativity and the possible binary expression trees that would evaluate to the same number?

**If a binary operator is commutative, then its left and right children (including their children/descendants/…) can be switched with no difference in the value resulting from the evaluation of the tree.**

14. (2 points)

Consider the following binary expression trees:



Mark up the tree on the left to indicate which nodes could be swapped while preserving the same output of the evaluation. To indicate this, draw an oval around two nodes which could be swapped (including their children and grandchildren etc…) while yielding the same result. As an example, in the tree to the right, Nodes B and E could be swapped, and Nodes F and G could be swapped.

**Any 2 of the 4 ovals marked up in red on the left tree.**

15. (1 points) What is the **Type** class used for in C#? How is it related to Boxing or Reflection or Delegates?

**The Type class stores information about a class. It is used with reflection to store type info.**

16. (1 point) circle the choice below that matches the binary expression tree above on the left.

1. 2\*3+4+(3\*4)/5
2. **2+3\*4+(3\*4)/5**
3. 2+3\*4/(3\*4)
4. None of the above

**Guidelines/Requirements for the coding question that follows**

* Full credit is given only for working C# code (points off for no pseudo-code or written explanations).
* Unreadable/messy code is worth 0 points. Don't have stuff scribbled out or arrows redirecting lines to different places. Implemented pseudo-code on scratch paper if need be to first figure out the solution approach, then implement the actual C# code on the problem page.
* Excessively lengthy implementations and/or implementations that involve strategies that are bad practice are worth few if any points.
* You may (and are actually encouraged to in order to keep your code simple and readable) create utility functions. Implement them on the back of the page if need be to keep your code clean.

22A. (4 points) Implement the Evaluate function for an expression tree class below. You may assume that the tree is correctly built and that m\_root references the root node in the tree. You may want to add one utility function to the class to assist with the evaluation.

**class ExpTree {**

**class Node {} class ConstNode { public double Value; } class VarNode { public string Name; }**

**class OpNode { public char Op; public Node Left, Right; }**

**Node m\_root; Dictionary<string,double> m\_vars = new Dictionary<string,double>();**

**public double Evaluate() {**

**return eval(m\_root)**

**}**

**private double Eval(Node n)**

**{**

**if n is const node**

**{**

**return constnode.value}**

**else if (n is varnode)**

**{**

**String name = (n as varnode).Name;**

**if (m\_vars.containsKey(name))**

**{**

**return m\_vars[name];}**

**else**

**return 0;}**

**}**

**return 0**

**}}**

**// Here is logic that they could NOT directly use, though if they had it they did show somehow**

**// that they knew it then they would get some partial credit. But more if they at least realized**

**// that the class structure above was different from HW5 in that there was no**

**// abstract Node class with a virtual double Eval() method, and then OpNode, ConstNode,**

**// and VarNode were children of Node that all overrode Eval() in their own way.**

**switch (operation) //since only first char matters (might have to change later with parenthesis)**

**{**

**case '+':**

**return this.left.eval() + this.right.eval();**

**case '-':**

**return this.left.eval() - this.right.eval();**

**case '\*':**

**return this.left.eval() \* this.right.eval();**

**case '/':**

**return this.left.eval() / this.right.eval();**

**}**

**}**

**}**