FAST NUCES – IPT – Assignment 1

Due: Sunday 24th September 2017

Please put ‘Assignment #1 <Reg #>’ in the subject line.

PLEASE DO NOT COPY

Part A

1. Create a custom class called DynamicIntArray that implements a dynamic array of integers.
   1. Keep a data member to store the current size of the array
   2. Keep a data member to store the current capacity of the array
   3. Keep a data member to store the actual array.
   4. Provide a default (no argument) constructor that sets the current capacity to 10 and allocates memory for a ten element array of integers.
   5. Provide another constructor that takes one argument to specify the initial capacity of array.
   6. Provide a method “void Add(int)” that appends an element to the end of the array. If the current size of the array reaches the capacity of the array, more memory is allocated to extend the capacity of the array. [HINT: Use Array.Resize() method]
   7. Provide a method “int Get(int)” that returns the element value of the index specified by the argument.
   8. Provide a method “int IndexOf(int)” that finds the element value specified by the argument and returns the index where it is found, or -1 if it is not found.
2. Write a program that compares the performance of the above Dynamic Array with C# array, ArrayList and List<int>.
   1. Populate the collections by generating 1M (one million) random values
   2. Carry out traversal and find the sum of the element values. Print the sum and the time required to carry out the traversal in each case.
   3. Search five randomly chosen values from each of the collection by calling the IndexOf() function and compare response times.

Part B:

1. Convert your DynamicIntArray class to a template/generic DynamicArray<T> class that can be an array of any type T.
2. Implement the IList<T> interface to the class.
3. Compare performance of DynamicArray<T> with List<T> and C# array for data types:
   1. Double
   2. Decimal
   3. Boolean

Grading Criteria:

Assignment will be graded according to the following general criteria:

* The solution adequately addresses the requirements for the assignment.
* The program compiles, links, and executes.
* The program runs correctly (or at least appears to be correct based on testing done by the grader).
* The program is easy to read and to understand, i.e., it is well commented and adheres to programming conventions. For example, method and object names are appropriate (or as defined by the assignment), and all potentially confusing/complex program code (including non-trivial if-statements, loops, method calls, etc.) is adequately documented.
* The general design of the program is clear and reasonable. For example, the program makes good use of classes and methods and is implemented in a sensible, understandable way.

Assignment will receive a grade according to the following table:

|  |  |
| --- | --- |
| Quality of Submitted Solution | Points (%) |
| The solution meets all criteria well | 100 |
| The solution meets most criteria, but there is some room for improvement | 80 |
| The solution is just satisfactory; it meets some criteria but there is significant room for improvement | 40 |
| The solution is not acceptable | 0 |

**Zero will be awarded for assignment that is copied (either amongst students or from another source).**