Assignments (50% of Final Course Grade) – COMP605

The aim of these four C# programs is to improve your programming skills and to give you a basic understanding of commonly used data structures and algorithms. Zip and upload each program to the relevant link in moodle. Name the file as: *FirstName_LastName_Assignment_(number)*

For marking:

1. You have to attend the lab to demonstrate your code
2. Answer questions
3. Apply simple modifications

Not attending the lab for marking will result in zero mark. I may deduct marks if you are unable to demonstrate and explain code or apply simple modifications. During classes, I will discuss these programs with you and give you more help and guidance. It is important that you attend all classes!

Marking guidelines:

- 50% - Correct run
- 30% - Clear design, indentation and comments
- 20% - Efficient design and simplicity
- 10% - Handling error scenarios

Assignment 1: Arrays and simple algorithms [12%]

Write a C# program that reads a text file (*inputassignment1.txt*) which includes 20 numbers (integers). Your program should read the numbers into an array and print to the screen the following:

1. The minimum number.
2. The position of the minimum number.
3. The number of prime numbers.
4. The average number.
5. The number of odds numbers
6. The number of even numbers
7. The number of numbers that are bigger than the average number.
8. A statement to indicate whether all numbers are equal or not.
9. A statement to indicate whether all numbers are different or not.
10. A statement to indicate if the numbers in the array are sorted or not.

Your program should include at least six different methods excluding the main method.

** This is a simple program and I expect all students to be able to write it without any problems. If you have problems I will help you during the labs.

** Can NOT use C# built in data structures.

** Build your own *inputassignment1.txt* file.

Due date: see moodle.

Assignment 2: Lists [13%]

Assume that *file1.txt*, *file2.txt* and *file3.txt* contain an unknown number of integers (text format) and assume that the numbers in each file are unique (different). Write a C# program that:

1. Reads the integers in file1.txt, file2.txt and file3.txt and inserts them into doubly linked lists *list1, list2* and *list3* respectively. Add the integers to the tail of the list.
2. Prints to the screen the size of each list.
3. Print to the screen the "Middle" number of each list.
4. Prints to the screen the prime numbers in each list (five numbers each line)
5. Prints to the screen the numbers in each list in reverse order.
6. Prints to the screen all the numbers that appear in the three lists (intersection)

****** No use of built in C# data structures and built in methods

** I will give you examples and help during the labs.

Due date: First term, see moodle.

Assignment 3: Binary search tree [12%]

Similar to assignment 2, assume that file1.txt, file2.txt and file3.txt contain an unknown number of integers (text format) and assume that the numbers in each file are unique. Write a C# program that:

1. Reads the integers in file1.txt, file2.txt and file3.txt and inserts them into binary search trees bst1, bst2 and bst3 respectively.
2. Prints to the screen the size of each tree.
3. Prints to the screen the numbers in each tree in decreasing order.
4. Prints the height of each binary search tree.
5. Prints the numbers in the binary search tree "level by level".
6. Print the number of prime numbers in the second binary search tree.

****** No use of built in C# data structures.

** I will give you examples and help during the labs.

Due date: Second term, see moodle.

Assignment 4: Hash table and Big O notation [13%]

The aim of this assignment is to implement and to apply the basic operations of hash table and to do basic running time analysis (Big o notation)

Part A – Coding [50 marks]

Assume that *file1.txt* contains an unknown number of strings (terms) (text format). Write a C# program that uses the built-in hash table and prints to the screen the term with maximum frequency.

****** You are allowed to use the internal (built-in) hash table implementation.

** Build your own *file1.txt* for testing.

Part B – Big O Analysis [50 marks]

Assume the sizes of file1.txt is (n)*(the number of words (strings) in each file is* $O(n)$*).*
Provide a detailed worst case analysis and average case analysis of your program.

Feel free to use the average case analysis and the worst case analysis of the operations of hash table as provided in the table below.

	Average case	Worst case
Insert	O(1)	O(n)
Search	O(1)	O(n)

Hash Table

** Help, clarification will be provided during the lectures/labs.

Due date: Second term, see moodle.