(FOR YOUR REFERENCE ONLY)

Workshop Test 1: Algorithm Analysis (Your marks: (out of 10))

Student ID: Name:

Tasks: Attempt all five questions below. Note: all workings must be shown!

1. (1 mark)
   Using a manual method, apply *floor*( ) and *ceiling*( ) functions to $log_2(1050)$.

2. (1 mark)
   Find the Greatest Common Divisor (GCD) of 1050 and 588 by manually executing the Euclid GCD algorithm shown on page 3 of the textbook (also see lecture01.ppt slide 7~9).

3. (2 mark)
   Suppose that the following expression is the sum of the time characteristic operations of an algorithm.
   $$776\times n^2\times\log_2(n)+3.1\times n^3+8\times n^2+\ 30\times n^{2/3}+850$$
   Determine the time complexity of the algorithm using the big-*O* notation.

4. (2 marks)
   Determine the time complexity of the following method, using *O*-notation.

```
int example(int[] array)       //line 01
{ if (array==null) return 0;   //line 02
  int n=array.length;          //line 03
  if (n==0) return 0;          //line 04
  int maximum=array[0];        //line 05
  int minimum=array[0];        //line 06
  for (int i=1; i<n; i++)      //line 07
  { if (array[i]>maximum)      //line 08
      maximum=array[i];        //line 09
    if (array[i]<minimum)      //line 10
      minimum=array[i];        //line 11
  }                            //line 12
  return n*maximum*minimum;    //line 13
}                              //line 14
```

5. (4 marks)
   Suppose we have a file **F** that contains $\mathbf{n^2}$ distinctive integer values, which are in ascending order. Consider the following process:
   - Consecutively, read the values from the file **F** and store them in the same order in an appropriately-sized array **A**.
   - Search the array **A** for a specific target value **T**.
   - Terminate with: either **+index** if **T** is found in cell **A[index]**, or **–1** if **T** is not found in array **A**.

   (1) Determine which "search algorithm" is best and state its steps.
   (2) Analyse the time complexity of the entire process in terms of *O*-notation.