**UBI602 Alorithmic Complexity Analysis**

**Homework 3**

**1.**Design a decrease-by-one algorithm for finding all the factors of a given number *n*. Also design a decrease-by-one algorithm for finding all the prime factors of a given number *n*.

**2.**Consider the following algorithm to check connectivity of a graph defined by its adjacency matrix.

**ALGORİTHM *Connected*(A[0..*n*-1,0..*n*-1])**

            // Input: Adjacency matrix A[0..*n*-1,0..*n*-1] of an undirected graph G

            // Output: 1 (true) if G is connected and 0 (false) if it is not

**if***n*=1 **return** 1                   // one-vertex graph is connected by definition

**else**

**if not** *Connected(*A[0..*n*-2,0..*n*-2]*)* **return** 0

**else for**j=0 **to** *n*-2 **do**

**if**A[*n*-1,*j*] **return** 1

**return**0

Does this algorithm work correctly for every undirected graph with *n*>0 vertices? If you answer yes, indicate the algorithm’s efficiency class in the worst case; if you answer no, explain why.

**3.**Consider the following implementation of the algorithm for generating permutations discovered by B. Heap.

**ALGORİTHM *HeapPermute*(*n*)**

            // Implements Heap’s algorithm for generating permutations

            // Input: A positive integer *n* and a global array A[1..*n*]

            // Output: All permutations of elements A

**if** *n*=1

**write** A

**else**

**for***i=*1 **to** *n* **do**

*HeapPermute(n-1)*

**if***n* is odd

            swap A[1] and A[*n*]

**else** swap A[*i*] and A[*n*]

a. Trace the algorithm by hand for *n* = 2, 3 and 4.

b. Prove the correctness of Heap’s algorithm.

c. What is the time efficiency of *HeapPermute*?