IELM 330: IT for Logistics and Manufacturing
Assignment 2, Max score= 5+5+5=15

Q1. Suppose that we want to sort an array of 'objects' in a VB 6.0 program. Each object is made up of some data, described in the code below:

```
Type SearchNode
    depth as integer         ' the current depth of our search
    jobs_done( ) as Integer ' an Array storing those jobs that we have already done
    weight as double
end Type
```

Please download the (incomplete) VB project files provided with this assignment, and complete the 'sort' function by implementing a simple bubble sort algorithm.

Your 'bubble_sort' function takes two inputs: (1) an array of SearchNodes, and (2) an integer storing the number of elements in input (1).
The function outputs an array that has all the elements of the input array, sorted in increasing value of the 'weight' of each node.

You must submit all the project files to the TA in a zipped file. The TA will use an arbitrary set of test data values to check whether your function is working properly.

Q2. Modify the lecture notes data for the scheduling example (you are free to change any numbers, and even add or delete jobs) such that neither EDD, nor SPT will result in the optimum schedule. [Note: both, EDD and SPT must fail to produce optimum schedule for the same set of jobs].

Q3. Suppose that we are faced with scheduling a very large number of jobs. Then our best-first method may take too long to find the optimum. Therefore, we want to try an alternate approach, called a 2-opt, as follows:

Step 1. Find a reasonable schedule by using EDD.
Step 2. Improve the schedule by repeatedly applying 2-opt, as follows:

Suppose we have an initial job sequence, like: j1 → j4 → j3 → j2.
We call it the *current best*, and compute the late cost of this schedule, say, L_0.
We consider each pair of jobs, and consider the late cost if we exchange the positions of this pair; e.g. exchanging j1 and j3 will give a schedule: j3 → j4 → j1 → j2, with a new late cost.
Among all the exchanged costs, whichever gives the maximum reduction in late cost, we make that switch, and call it the *current best.*
Repeatedly apply 2-opt to the current best, until either (i) no pair-wise exchange can improve the solution, or (ii) the program has run or over 10 minutes.

Write a detailed ***pseudo-code*** of this method, called function, *sched_edd_2opt ()*.