Exercise 1 (Course questions) (06 points)

1) **What does a decision problem mean?**
2) **Give two mathematical relations showing how the complexity classes P, NP, NP-complete, and NP-hard are related.**
3) **In a combinatorial optimization problem that consists to minimize a function *f* on the search space S, what must satisfy an optimal solution x*?**
4) **Given two algorithms solving the same problem: A1 with time complexity $O(n^2)$ and A2 with time complexity $\Theta(n^2)$, which one would you choose? Justify your answer.**
5) **Why do we use a memorization table in dynamic programming?**
6) **When becomes B&B most efficient?**

Exercise 2 (05 points)

Consider the symmetric TSP with n=4 cities A, B, C, D such as AB=8 , AC=4, AD=2 , BC=7, BD=6, CD=4.

1) **Compute the length of the tour ACDB.**
2) **How many possible solutions (tours) are there? (write the rule).**
3) **Starting from city A, determine a tour t using the Nearest Neighbor heuristic then compute its length. Does t represent a lower bound or an upper bound of the optimal solution? Justify your answer.**
4) **Write a heuristic computing the other bound (just the idea, no need to write the algorithm).**

Exercise 3 (09 points)

The algorithm bellow takes as input an integer array c and an integer A, and it provides as output an integer array x and an integer T.

1) **a) Compute x and T for c = [10 , 5 , 2 , 1] and A= 36.**
   b) Express the time and space complexities of this algorithm in O notation.
2) **a) Which combinatorial optimization problem this algorithm solves?**
   b) Write the mathematical formulation of this problem.
   c) Determine the search space and its size.
3) **What is the paradigm of this algorithm (how is it called).**
4) **a) Write a dynamic programming algorithm that solves the same problem.**
   b) Without computing again, compare the value T_{DP} of T provided by dynamic programming algorithm and T_1 who's obtained in question (1). (Justify your answer).

```
Algo( int[] c, int A){
Sort c by descending order
n = length(c);
T = 0;
for (int i = 0; i < n; i++)
     x[i]=0;
     while (c[i] <= A)
               { A - = c[i];
                  x[i] + =1;  }
      T + = x[i]  }
return x[], T; }
```