`LAB: Polymorphism` Group Assignment

Objective: The goal is to introduce an important feature of OO Programming, namely Polymorphism. We shall appreciate this feature by taking case studies that follow Dynamic programming or Greedy Technique.

Motivation: 1 Is there a succinct way of implementing functions with similar operations. For example, finding max and second max in a set of integers, in a set of floating point numbers, in a set of characters (characters are compared based on their ascii value). For all three operations, the common theme is 'find max and second max' and the only change is 'data type'. We are looking for a function doing multiple roles (poly + morph).

Answer: Function Templates.

Task Expected out of you: Read Function templates and implement the following;
(i) Find max and second using $n + \log n - 2$ comparisons (check problem session), if

(ii) Find max and min with the least number of comparisons (check problem session), if

- the input is an array of integers
- the input is an array of double
- the input is an array of characters
- use functiontemplate.h to define template and in .cpp invoke the function with appropriate data type as an argument. Do NOT forget to include .h file at the beginning of the program (include "functiontemplate.h").

Motivation: 2 Is there a succinct way of implementing functions with similar operations accepting different number of arguments. **Answer:** Function overloading.

Task Expected out of you: Read function overloading and implement the following;

- Knapsack(int W, int Weight, int Profit)
  - Implement Dynamic programming based approach, to be discussed in class on Monday.
- Knapsack(double W, double Weight, double Profit)
  - Same as above, however, in this we accept floating point numbers.
- Knapsack(int W, int Weight)
  - Implement `knapsack-variant:` Knapsack problem with no profits. That is, the objective is to find a subset of objects with maximum cardinality. Implement DP/Greedy. `Hint:` Container Loading.
- Knapsack(int W, int Weight, int Profit, bool flag)
  - In all of the above, we were not allowed to pick a fraction of objects into the Sack. Suppose, we are allowed to pick a fraction of objects, i.e., $(x_1 = 1, x_2 = 1, x_3 = 0.5)$ such that $w_1 + w_2 + 0.5w_3 = W$. Note: the sum equals W as we are allowed to pick a fraction now. 'flag' is artificially introduced ('flag' is set to true always) so that one can distinguish between fractional knapsack and non-fraction knapsack problem. Implement Greedy Strategy 3 which was discussed in class.

Note:
0. Let there be a good reading before attempting to code these questions.
1. main() can not be over loaded.
2. int Knapsack() and double Knapsack() – not permitted. Compiler distinguishes the function based on the name, the number of arguments, and the data type. It does not distinguish based on 'return type'.
3. Please discuss with me, if you have doubts.