Lab 8: Recursin

Data Structures and Algorithms

Today, we will a series of recursion problems.

The following program recursively print out all element of array a if you call printArray(a,0)

```
static void printArray(int a[], int i) {
  if(i==a.length) return;
  System.out.print(a[i]+" ");
  printArray(a,i+1);
}
```

Task 1: modify printArray to print the array in reverse order, call it printReverse

Task 2: modify printArray to print the array and its reverse order after it. For example, if a = {1, 3, 4, 7} the printout will be 1 3 4 7 7 4 3 1, call it printArrayAndReverse

Enough warm up!

There is a classic problem call 0-1 Knapsack Problem. The problem goes like this [definition and image from https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/].

Given weights and values of n items, put these items in a knapsack of capacity W to get the maximum total value in the knapsack. In other words, given two integer arrays value[0..n-1] and weight[0..n-1] which represent values and weights associated with n items respectively. Also given an integer W which represents knapsack capacity, find out the maximum value subset of value[] such that sum of the weights of this subset is smaller than or equal to W. You cannot break an item, either pick the complete item or don't pick it (0-1 property).

0-1 Knapsack Problem

value[] = {60, 100, 120};
weight[] = {10, 20, 30};
W = 50;

Solution: 220

Weight = 10; Value = 60;
Weight = 20; Value = 100;
Weight = 30; Value = 120;
Weight = (20+10); Value = (100+60);
Weight = (30+10); Value = (120+60);
Weight = (30+20); Value = (120+100);
Weight = (30+20+10) > 50

Task 3: You are to implement the 0-1 knapsack problem with a twist. The input and other requirement are the same with the knapsack problem. However, there are two more constrains:

1. Number of items in the knapsack must be even.
2. The heaviest (highest weight) item and the lightest (lowest weight) item must not be in the knapsack at the same time. You can only pick one to put in the knapsack.

For example

```
Input:
  Index   = [  0, 1,  2,  3,  4,  5 ]
  value   = [ 20, 5, 10, 50, 40, 25 ]
  weight  = [ 1,  2,  3,  8,  7,  4 ]
  int W = 10

Output: Knapsack value is 60
index  :  0, 1,  2,  5
value  : 20, 5, 10, 25
weight :  1, 2,  3,  4
```

Note that a set of indexes {0,3} which has a value of 70 is not possible since the heaviest and the lightest item are both in the bag(constrain 1). Also, the set of indexes {0,1,4} which has a value of 65 is also not possible since it contains odd number of items (constrain 2).

Instruction: For task 1 and 2, implement printReverse, praintArrayAndReverse as static method together with main method for testing them. Put all the code in a file call Lab8Task1.java for submission. For task 3, implement your solution in a file name Lab8Knapsack.java for submission.