1. (40 pts.) Snakes on a tree.

Let $G = (V, E)$ be an undirected graph with weights on the edges. That is for an edge $uv \in E$, we have a weight $w(uv)$ associated with $uv$. A $k$-snake in a graph is a simple path with exactly $k$ edges in it. Two $k$-snakes are disjoint if they do not share any vertex. A collection $M$ of $k$-snakes is a $k$-snake packing if all pairs of snakes of $M$ are disjoint (for $k = 1$ the set $M$ is a matching in the graph). The total weight of a $k$-snake packing is the total weight of the edges used by the snakes.

We are interested in the problem of computing the maximum weight $k$-snake packing in $G$. In general, this problem seems hard. Fortunately for the tree case this is much easier.

Describe an efficient algorithm (i.e., provide pseudo-code, etc), as fast as possible, for computing the maximum weight $k$-snake packing when $G$ is a rooted tree. Your algorithm should be recursive and use memoization to achieve efficiency. (You can not assume $G$ is a binary tree - a node might have arbitrary number of children.) What is the running time of your algorithm as function of $n = |V(G)|$ and $k$?)

For example, the following shows a tree with a possible 3-snake packing.

2. (30 pts.) All your internet base are belong to us.

You are given a road somewhere out there. The road is straight, and there are $n$ farm houses along it. You are given the distance $x_i$ of the $i$th house from the beginning of the road. (You can assume $x_1 \leq x_2 \leq \ldots \leq x_n$.) You need to place $k$ stations along the road, such that you connect every house to one of the stations. The price of connecting a station to a house is their distance.

(A) Prove that if $n$ is odd and $k = 1$ then the optimal placement for the single station is at $x_{(n-1)/2+1}$.

(B) Prove that if $n$ is even and $k = 1$ then the optimal placement for the single station is at either $x_{(n)/2}$ or $x_{(n)/2+1}$ (and both locations have exactly the same price).

(C) Describe an algorithm, as fast as possible, just that given $x_1, \ldots, x_n$, and an integer $k$, outputs the locations of the best placements for the $k$ stations (sorted by increasing order), and the price of wiring all the houses to these stations. How fast is your algorithm? (specifically, what is the running time if $k = O(1)$ or $k = n/2$.)
3. (30 pts.) And you thought this was a theory class.

Implement your algorithm from the previous part in C or C++. The input to your program will be

\[
\begin{array}{c}
  n \\
  k \\
  x_1 \\
  : \\
  x_n
\end{array}
\]

(Given on the standard input.) To simplify things you can assume all these numbers are integers. The output should be:

(A) Price of the solution (on its own line).
(B) The location of the \( k \) stations (each on its own line) – in increasing order.
(C) For each house, the index of the station (out of the \( k \) stations) it is connected to (each on its own line).

You need to submit a single C (or C++) file that contains your program (which must compile on its own and run efficiently). We will provide several input/output examples on the class webpage. We would also provide some help code to implement memoization.\(^1\)

\(^1\)We will use MOSS to verify that programs are not copied...