Input File: `pearls.in` **100 Points**

Output File: – **Time limit:** 1 s

Source File: `pearls.pas/.c/.cpp` **Memory limit:** 16 MB

Solution

The tasks is about finding winning moves. A winning move is a move which leads the player to a win, if he plays optimal in the following moves.

If we suppose that both players play optimally, each move is either a winning move or a losing move as there is no draw in this game.

A move is further a winning move, if a dwarf can give the necklace to a dwarf of his tribe who has a winning move, or if a dwarf can give the necklace to a dwarf of the opposite tribe who does not have a winning move.

A game situation is determined by the state of the necklace and the current dwarf:

$$\begin{aligned} &hasWinningMove(c_1c_2\ldots c_k, d) := \\ &\quad \exists d' \in d.list(c_1):\ d.color = d'.color \wedge hasWinningMove(c_2\ldots c_k, d') \quad \vee \\ &\qquad\qquad d.color \neq d'.color \wedge \neg hasWinningMove(c_2\ldots c_k, d') \end{aligned}$$

$hasWinningMove(c_1, d)$ is always true, as there is only the diamond left.

With this information you can easily compute who is going to win and which moves to take, by using bottom-up dynamic programming. The dimensions of the dynamic programming matrix are the indices in the necklace $(1\ldots L)$ and the dwarfs $(1\ldots N)$. The main loop is over the necklace indices in reverse order, i.e. $L,\ldots,1$.