АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
ИГРА ПО ДЪРВО

Съставяме графа, като отбелязваме кои върхове са листа и каква е тяхната стойност съответно в масивите is_leaf[] и v[]. За да решим задачата е необходимо да пуснем обхождане в дълбочина от корена на дървото към листата. При самото обхождаме ще пресмятаме дали ние, намирайки се в текущия връх, ще спечелим играта, ще я загубим или тя ще свърши с равенство. За целта разглеждаме неговите наследници. Ако отивайки в някой от тях, опонентът ни ще загуби, то ние печелим. Ако няма такива, но има наследник, който води до равенство, то нашата игра ще свърши с равенство. В противен случай ще загубим.

Специфично е единствено достигането на листа, тъй като информацията в тях е по-различна от тази, която връща обхождането. Затова към стейта на dfs-то добавяме и променливата player, която да пази кой е играчът, който е на ход в текущата ситуация. Затова ако например достигнем листо със стойност „-1“ и играчът, който е на ход е вторият, връщаме, че текущата ситуация е печеливша. Аналогично за останалите възможности.

Кодът на програмата:

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <vector>

using namespace std;

int N;

bool is_leaf[1000];
int v[1000];

vector <int> n[1000];

int dfs (int u, int player) {
   if (is_leaf[u]) {
      if (player == 0)
         return v[u];
      else
         return v[u]*(-1);
   }

   int res = -1;

   for (int i = 0; i < n[u].size(); i++)
      res = max (res, dfs (n[u][i], player^1)* (-1));

   return res;
}
int main () {
   scanf ("%d\n", &N);
```

```
    char type;
    int an;

    for (int i = 1; i < N; i++) {
        scanf ("%c", &type);

        if (type == 'L') {
            scanf (" %d %d\n", &an, &v[i]);
            is_leaf[i] = 1;
            an--;
            n[an].push_back (i);
        } else {
            scanf (" %d\n", &an);
            an--;
            n[an].push_back (i);
        }
    }

    int ans = dfs (0, 0);
    if (ans > 0)
        printf ("+");

    printf ("%d\n", ans);
}
```