Решение:

Структурата данни, в която се пази дървото на папките от А използва масивите `int a[N][M], at[N]; char as[N][10];` В `at[j]` е записан съответният момент време за `j`-тата папка (или файл), в `as[j]` – името й. Броят на наследниците й се съхранява в елемента `a[j][0]`, а номерата на наследниците – в `a[j][1]`, `a[j][2]`, ..., `a[j][a[j][0]]`. В `na` е общият брой на всички папки и файлове от А. По аналогичен начин се изгражда структурата от данни за В, като се ползват съответно `b[N][M], bt[N], bs[N][10]` и `nb`. Запълването на тези данни се извърша от фукцията create(), извиквана два пъти, за да обработи входния файл за А и за В.

Функцията `compare(int n)` рекурсивно и синхронно обхожда двете дървовоидни структури, спускайки се в дълбочина. Всеки път, когато се достигне връх на дървото А, при което се открива, че съответния връх на В няма необходимите свойства за "еднаквост" (например върхът в А е папка, а този в В – е файл, или двата върха са файлове, но този в А е с по-ново време), навлизането в дълбочина се прекратява и евентуално се извиква рекурсивната функция `count()`, която преброява файловете от поддървото на А, което започва от въпросния връх. Променливата `c` служи за глобален брояч, чиято стойност се отпечатва накрая на програмата. Във функцията `main()` се обработва коренът на дървото, който има номер 1.

```
//ANSI C++
#include<iostream>
#include<cstring>
using namespace std;

const int N=999;
const int M=19;
int a[N][M], at[N]; char as[N][10];
int b[N][M], bt[N]; char bs[N][10];
int n,na,nb;
int c=0;

void create(int a[N][M], int at[N], char as[N][10])
{
 cin >> as[n] >> a[n][0] >> at[n];
 int n0=n;
 for(int i=1; i<=a[n0][0]; i++)
  {
    n++;a[n0][i]=n;
    create(a, at, as);
  }
}

void count(int n)
{
 if(a[n][0]==0) c++;
 else
 for(int i=1;i<=a[n][0];i++) count(a[n][i]);
}

void compare(int n)
{
 int i,j,j0;
 for(i=1; i<=a[n][0]; i++)
  {
    int f=0;
    for(j=1; j<=b[n][0]; j++)
```

```
     if(strcmp(as[a[n][i]],bs[b[n][j]])==0) {f=1; j0=j;}
    if(f==1)
    {
     if(at[a[n][i]]==bt[b[n][j0]])
      {
        if((a[a[n][i]][0]!=0)&&(b[b[n][j0]][0]!=0)) compare(a[n][i]);
        else if((a[a[n][i]][0]==0)&&(b[b[n][j0]][0]!=0)) c++;
        else
         if((a[a[n][i]][0]!=0)&&(b[b[n][j0]][0]==0)) count(a[n][i]);
      }
     if(at[a[n][i]]>bt[b[n][j0]]) count(a[n][i]);
    }
    if(f==0) count(a[n][i]);
  }
}

int main()
{
 n=1;create(a, at, as);na=n;
 n=1;create(b, bt, bs);nb=n;
 if(strcmp(as[1],bs[1]) != 0) count(1);
 else if((a[1][0]==0)&&(b[1][0]!=0)) count(1);
 else if((a[1][0]!=0)&&(b[1][0]==0)) count(1);
 else if(at[1] > bt[1]) count(1);
 else compare(1);
 cout << c << "\n";
 return 0;
}
```