Soluzione del II Compitino dell'11/06/2021

```
void matchF(nodo*T, int*P,int &im, int dimP)
{
 if!T || im == dimP) return;
 if(!T->left&& !T->right)
   {
     if (T->info==P[im])
      {
        L[im]=T;
        im=im+1;
      }
   }
else
   {
      matchF(T->left,P, im, dimP);
      matchF(T->right, im, dimP);
   }
}
```

PRE=(alb(T) è ben formato 0<=im<=dimP, P ed L hanno dimP posizioni, vim=im)

POST=(0<=im<=dimP, L[vim..im-1] contiene puntatori alle foglie di T che matchano P[vim]..P[im-1] in modo che elementi successivi di L puntano a foglie successive di T (da sinistra a destra))

La prova di correttezza:

Caso base : T=0 oppure im = dimP non fare niente rende vera la POST perché o non ci sono nodi in T oppure non ci sono elementi di P da considerare.

Caso base bis: T è una foglia:

i) se P[im]=T->info allora L[im]=T; im=im+1; rende vero POST perché im=vim+1 e quindi L[vim..im-1]=L[vim] che infatti punta a T che contiene info=P[vim].

ii) altrimenti, basta fare return per soffisfare la POST infatti L non cambia visto che T è una foglia che non match P[vim].

Caso ricorsivo:

PRE_ric_1 è vera per la PRE infatti im non è cambiato e quindi è ancora vero che 0<=im<=dimP. Vale la POST_ric_1 e quindi L[vim..im-1] contiene puntatori alle foglie del sottoalbero sinistro di T che matchano P[vim..im-1]. Chiamiamo vim2 il valore di im dopo la prima invocazione. Quindi la parte di L riempita dall'invocazione a sinistra è L[vim..vim2-1].

Anche la PRE_ric_2 è vera, ma ora dobbiamo notare che la POST_ric_1 garantisce che vim2<=dimP. Quindi, ragionando come prima, vediamo che l'invocazione a destra riempie L[vim2..im-1] con puntatori alle foglie del sottoalbero destro di T che matchano P[vim2..im-1]. Quindi alla fine delle 2 invocazione L[vim..im-1] contiene i puntatori alle foglie di alb(T) che matchano P[vim..im-1].