Esercizio 2

L'esercizio riguarda il match di un pattern sui cammini di un albero binario. Il match deve essere completo, ma non necessariamente contiguo: vediamo un esempio.

Esempio: supponiamo che l'albero in questione sia il seguente:

1(2(4(8(_,_),_),6(0(_,_),_)),3(5(9(_,_),_),7(_,_)))

e che si cerchi un match di P=[1,3,7] su di esso. Allora il match coinvolge la radice, il suo figlio destro e il nipote all'estrema destra dell'albero:

1(2(4(8(_,_),_),6(0(_,_),_)),**3**(5(9(_,_),_),**7**(_,_)))

L'esercizio, in caso di successo del match, chiede di produrre una lista concatenata con 3 nodi il cui primo nodo ha campo info che punta alla radice, il secondo nodo punta al figlio destro della radice e il terzo nodo punta al figlio destro di quest'ultimo.

Si osservi nel programma dato che il tipo nodo delle liste concatenate ha campo info di tipo `nodoA*` per poter puntare ai nodi dell'albero.

A parità di albero, se P fosse [4,10], non esisterebbe alcun match sull'albero e in questo caso il programma dovrebbe stampare "`no match found`".

Si chiede pertanto di scrivere una funzione ricorsiva `match` che rispetti la seguente specifica:

```
PRE=(albero(r) ben formato, P ha dimP elementi con dimP>0)

nodo* match(nodoA* r, int* P, int dimP)

POST=(se esiste su un cammino di albero(r) un match di P, allora,
restituisce una lista concatenata con dimP nodi ciascuno dei quali punta
ad un nodo di albero(r)  che partecipa al match trovato)&&(altrimenti
restituisce 0)
```

I nodi puntati dalla lista restituita corrispondono al primo match trovato percorrendo l'albero in **ordine prefisso**.

Correttezza: dimostrare la correttezza di `match` rispetto a `PRE` e `POST` date.