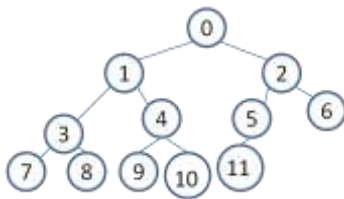



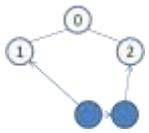
**Parte iterativa:** Si tratta di costruire un albero per strati, cioè prima la radice, poi i figli, poi i nipoti e così via. L'idea è quella di avere sempre una lista di puntatori ai nodi dello strato corrente in modo da aggiungere ad ogni nodo di quello strato due figli che faranno parte del prossimo strato e quindi dovranno essere inseriti in una seconda lista che diventerà il nuovo strato corrente quando le aggiunte a quello attuale finiscono, cioè tutti i suoi nodi hanno ricevuto 2 figli. I campi info dei nodi che vengono inseriti nell'albero vengono letti dal file "input" e il processo finisce quando questi valori finiscono.

**Esempio:** supponiamo che il file "input" contenga 12, che specifica quanti nodi in totale dobbiamo creare, seguito dai seguenti 12 valori 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 che serviranno per inizializzare i campi info dei 12 nodi da creare. L'albero da costruire con questi 12 valori è il seguente:

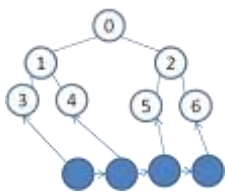


Vediamo i vari passaggi. Si parte dalla sola radice che corrisponde alla situazione seguente:

 in cui il pallino blu rappresenta una lista che punta ai nodi dell'albero. In questo caso solo la radice. I nodi blu hanno campo info di tipo nodo\*, cioè puntatore a nodo dell'albero. Usando la lista blu, aggiungiamo ad ogni nodo dell'albero puntato dai nodi blu, 2 figli e contemporaneamente dobbiamo costruire un'altra lista blu che punta ai nodi appena aggiunti all'albero (da sinistra a destra nello strato). La situazione diventa la seguente:



la precedente lista blu è consumata e questa diventa la nuova lista blu. Dopo avere processato e consumato questa lista blu, arriveremo nella situazione:



e così via per i 5 nodi che ancora mancano al prossimo livello per ottenere l'albero

finale dato prima. Quindi le operazioni su questo strato termineranno dopo ulteriori 5 letture anche se lo strato non viene consumato completamente. Si osservi che il terzo nodo dello strato corrente (quello con info=5) riceverà solo il figlio sinistro (con info=11) visto che a questo punto avremo creato 12 nodi come specificato dal primo valore su "input".

Per gestire le liste blu, nel programma che vi viene fornito, trovate la struttura ni e anche la struttura dni che possiamo rappresentare nel modo seguente:



le operazioni che servono sono `pend_ni` e `pop_ni`, dove la prima aggiunge un nodo `ni` alla fine della lista gestita da un `dni` e la seconda elimina il primo nodo della lista gestita da un `dni`. Le 2 funzioni sono `date`.

E' richiesto di realizzare una funzione iterativa `dni creastrato(int& n, dni X0, ifstream & IN)` che, data una lista blu gestita dal `dni X0`, aggiunge figli ai nodi dell'albero puntati da questa lista leggendo i campi `info` di questi nodi da `IN` (stream di "input") e restituisce una nuova `dni` che gestisce la lista blu che punta ai nodi aggiunti. Il parametro `n` indica quanti valori sono ancora disponibili su `IN`. La funzione deve soddisfare le seguenti specifiche:

**PRE**=( $n > 0$  ( $vn$  è il valore iniziale), `X0` è lista corretta non vuota i cui nodi puntano a nodi di un albero (sia `vX0` il suo valore iniziale), `IN` contiene almeno  $n$  valori)

**POST**=(aggiunge  $x = \min(vn, 2 * \text{lunghezza}(\text{lista gestita da } vX0))$  nodi all'albero, i campi `info` di questi nodi sono letti da `IN` e restituisce un `dni` che gestisce una lista blu di  $x$  nodi che puntano agli  $x$  nodi dell'albero appena creati)&&(la lista gestita da `X0` è deallocata e  $n$  ha valore  $vn-x$ )

**Correttezza:** definire l'invariante del ciclo principale della funzione `creastrato`.

**Parte ricorsiva:** usare ancora la struttura `dni` e le sue funzioni per ottenere una lista di nodi blu che puntano a tutti i nodi dell'albero costruito dalla parte iterativa dell'esercizio nell'ordine corrispondente ad una visita infissa<sup>1</sup>.

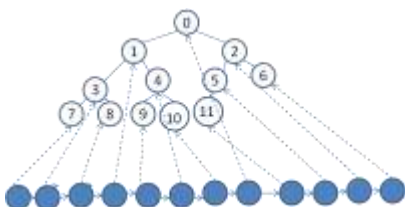
La funzione da fare deve soddisfare la seguente specifica:

**PRE**=( $r$  è albero binario corretto)

`dni unfold(nodo* r)`

**POST**=(la funzione restituisce un `dni` che gestisce la lista di nodi blu che punta a tutti i nodi dell'albero radicato in  $r$  nell'ordine corrispondente ad un attraversamento infisso dell'albero)

**Esempio:** considerando l'albero finale dell'esempio precedente la lista di nodi blu che va costruita da `unfold` è la seguente:



**Correttezza:** dare la dimostrazione induttiva della funzione `unfold` rispetto a pre e post date.

Osservate che nel programma dato c'è un `main` che invoca le 2 funzioni da fare e anche le funzioni che producono gli output attesi. Queste funzioni sono `date`. E' anche data la funzione `void svuota(dni)` che serve a deallocare i nodi di una lista blu gestita da una struttura `dni`.

<sup>1</sup> L'ordine infisso significa: prima il sottoalbero sinistro, poi la radice e poi il sottoalbero destro.

