Esercizio n.1

Si chiede di definire una funzione ricorsiva `buildtree(nodo*& R, int dim)` che legga `dim` valori (`dim` > 0) e costruisca un albero binario `R` con `dim` nodi il cui campo `info` contenga i valori letti e che segua la seguente modalità di costruzione dell'albero:

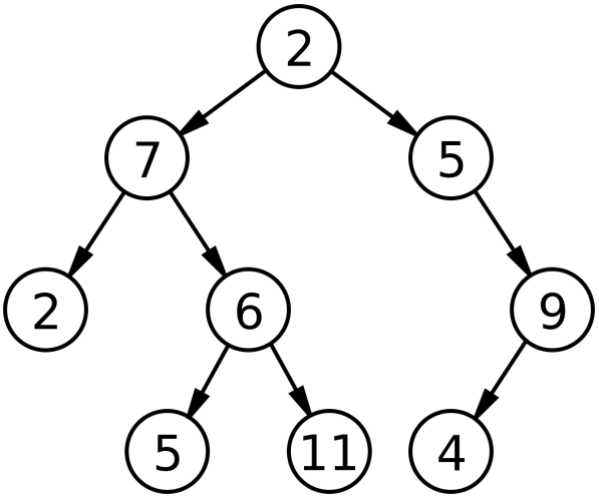
i) Inizialmente l'albero `R` è vuoto;

ii) Si legge il prossimo intero `x` e si inserisce un nuovo nodo con `info=x` nell'albero in modo da ottenere sempre un **albero bilanciato**. Per farlo, nel caso che l'albero abbia una radice, si sceglie di inserire il nuovo nodo nel sotto-albero (left/right) che abbia meno nodi. In caso di parità, si inserisce il nodo nel sotto-albero sinistro.

Si richiede anche di scrivere la funzione `stampa_l` che stampa un qualsiasi albero binario in formato lineare. Questa funzione è stata vista in classe ed è nelle slide.

Una volta costruito l'albero bilanciato, lo si deve percorrere in **ordine prefisso**, stampando alcuni campi `info` dei nodi attraversati: infatti si deve stampare solo 1 nodo ogni `k` nodi attraversati, per un dato `k>0`.

Esempio. Se l'albero da attraversare è il seguente:



Allora l'ordine prefisso dei suoi nodi è:

2 7 2 6 5 11 5 9 4

Se `k=2` allora dovremo stampare: 7 6 11 9. Se `k=4` stamperemo invece: 6 9.

Queste stampe devono essere eseguite da una funzione ricorsiva `stampa_a_salti` che rispetta le seguenti specifiche:

```
PRE=(albero(r) è ben formato, k>0, 1<=n<=k)

int stampa_a_salti(nodo* r, int k, int n)

POST=(percorre i nodi di albero(r) in ordine prefisso stampando i
campi info di alcuni di essi nel modo seguente: salta i primi n-1,
```

`stampa il campo info del nodo n-esimo, poi ne salta k-1 e stampa quello successivo e così via fino a visitare tutti i nodi di albero(r)) && (restituisce un intero m (1<=m<=k) che indica che si dovranno saltare m-1 nodi prima di stampare il prossimo)`

Per capire l'ultima parte della `POST`, consideriamo di nuovo l'albero di figura e assumiamo che `k=4`. Come detto prima, l'invocazione di `stampa_a salti` sulla radice dell'albero (2) con `k=4` e `n=4`, stamperebbe 6 e poi 9 in quanto dovrebbe non stampare i primi 3 nodi (in ordine prefisso) dell'albero e poi stampare il quarto nodo (6), poi non stampare i successivi 3 e stampare il successivo 9. A questo punto, l'albero ha solamente 1 nodo ancora da visitare (che quindi non verrà stampato!). Tuttavia, per garantire che al chiamante della funzione sia noto che il prossimo nodo da considerare vada stampato, la funzione *ad un certo punto* dovrà restituire `m=1`.

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