Second module: final exercise

Complete the 1D SPH code discussed in the last lesson of the Course. Use a binary tree for the neighbour search.

To this aim, you can implement a function that counts the number of neighbours in the interval [left_bound,right_bound] inside a tree-node:

```
int rangeCount(struct node* node, float left_bound, float right_bound):
if node == NULL: return 0
else:
        me = ,,1 if node's particle.Pos is within [left_bound,right_bound], else 0''
        return rangeCount(node->left, left_bound, right_bound) +
                rangeCount(node->right, left_bound, right_bound) + me;
```

This function can be used to determine the needed space for the allocation of a neighbour array. Then, you can write a function that, given an array (struct node *ngbs) allocated to contain n_ngbs neighbour, fills it with a copy of nodes within the search range:

```
int rangeSearch(node, left_bound, right_bound, ngbs, n_ngbs):
if(node == NULL): return 0;
else:
        int left = rangeSearch(node->left, left_bound, right_bound, ngbs, n_ngbs);
        int right = rangeSearch(node->right, left_bound, right_bound, ngbs,
                                        n_ngbs - left);
me = ,,1 if node's particle.Pos is within [left_bound,right_bound], else 0''
if me==1:
        ngbs[n_ngbs - (1 + left + right)] = *node; #fill neigbours list backward

return me + left + right; #returns how many neighbours have been filled
```

This is however just a hint, the search can be done in any way that uses a binary tree. Evolve the shock tube until you see all the 5 regions described during the Course.