# Problem 8

We can modify the AVL tree to achieve the goal.

We can construct a AVL tree normally, and bound a variable called 'last_insert' with it. The 'last_insert' stores the key of the most recently inserted pair. We need to make sure this variable valid and up-to-date by modifying it during the normal inserting process (i.e. change it into the new key when insert a new pair).

For the function deleteLast(), actually, we just need to call function delete() with the parameter 'last_insert'.

Since the normall AVL tree has the running time of search, insert, and delete all in $\Theta(\log n)$, we did not change the funciton of search and delete, so the running time of our own search and delete will not change. For insert(), what we do is just add a constant time consuming statement into the function, which keeps the insert() still in $\Theta(\log n)$. For deleteLast(), what we do is just call another function with the running time $\Theta(\log n)$, which implies that the running time for deleteLast() is also $\Theta(\log n)$.

And the additional space we used compared to the AVL tree is $\Theta(1)$, which satisfied with the requirement in the question.