Esercizio 2 del 5/5/2020

Questo esercizio è una continuazione dell'esercizio 1. Si vuole affrontare un problema che molto probabilmente la vostra soluzione dell'Esercizio 1 presenta e cioè, il problema di creare molte liste che vengono confrontate in modo da scegliere quella a valore massimo. Probabilmente però le liste costruite e non scelte vengono semplicemente lasciate nello heap, in quanto il loro nodi sono costruiti con delle new, ma la soluzione non si cura di deallocare con opportune delete i nodi delle liste che non sono quella finale, cioè quella che corrisponde al migliore cammino trovato.

Nel file che viene dato con questo esercizio, sono definite 2 variabili globali, countNew e countDel, che hanno il compito, rispettivamente, di contare quante operazioni di allocazione di nodi vengono fatte e quante deallocazioni di nodi. Vi viene chiesto di modificare la soluzione dell'Esercizio 1 in modo da aumentare countNew ad ogni new nodo e di aumentare countDel ad ogni delete nodo. Alla fine del main i valori delle 2 variabili globali vengono stampati. Ovviamente se tutte le operazioni di delete possibili sono eseguite, countNew-countDel=n in quanto la migliore lista ha lunghezza n.

Questo esercizio è istruttivo in quanto rivela quanto sia facile riempire lo heap di dati inutili. Fare questo è considerato un errore (memory leak) in quanto si rischia di riempire inutilmente lo heap causando il possibile fallimento di programmi apparentemente corretti.

I test per questo esercizio sono gli stessi di quelli dell'Esercizio 1, estesi con le stampe delle 2 variabili globali.