In questo breve programma voglio riassumere i principali comandi che è bene ricordare quando bisogna lavorare con le liste e con i loro puntatori.

```
# include<stdlib.h>
# include<iostream>
using namespace std;
typedef struct dato * np;
typedef struct dato { int x; np p;} elem;
```

/ voglio abbreviare la scrittura delle strutture e dei loro puntatori con il comando typedef. Da ora in avanti potremo chiamare i puntatori con la parola (decisamente piu breve) np e la relativa struttura con elem.*/*

//Questo programma crea una lista decrescente da 10 no a 1, //inserendoli ricorsivamente dal primo all'ultimo.

```
np inlista(int n) //n è il num elem lista, ritorna punt al primo elemento
{
if(n==0) return NULL;       // n=0 solo quando fine della lista
np punt;
punt=new elem;                    //oppure punt=new struct dato;
punt->x=n;      // oppure anche (*punt).x=n; come visto a lezione
punt->p=inlista(n-1);   //così attacco a punt il resto della lista
return punt;          //attacco punt all'elemento che lo precede.
}

main()  {
np temp,q;
temp=inlista(10);
for (q=temp; q!=NULL; q=q->p)
cout<<q->x;
}
```

Un esercizio interessante a questo punto è vedere che succede se nell'ultima istruzione sostituisco la riga `cout<<q->x;`con `cout<<q->p;` Ecco l'output che ho ottenuto facendo questa piccola modica:
6299696 6299728 6299760 6299792 6299824 6299856 6299888 6299920 6299952 0
Che cos'è questo? Il programma ha semplicemente stampato gli indirizzi di memoria in cui ha salvato gli elementi della lista. Notare che l'ultimo elemento (cioè NULL) viene stampato come un semplice zero (NULL==0, come potete facilmente vericare).