Esame di Programmazione 1 30 agosto 2006

Il tempo a disposizione è 2 ore.

Parte iterativa: consiste di 2 esercizi indipendenti

1) Il problema riguarda l'ordinamento di un array di interi effettuando scambi tra elementi contigui. Consideriamo un esempio. Sia A=[3, -1, 0, 8, 5] l'array da ordinare, la tabella d'inversione di A è l'array B=[2,0,0,1,0] dove B[i] è il numero di elementi minori di A[i] che si trovano alla sua destra, cioè in A[i+1..4]. La somma degli elementi di B è il numero minimo di scambi tra elementi contigui necessario per ordinare l'array.
   Si chiede di scrivere una funzione int * TAB(int *A, int dimA) che costruisca la tabella d'inversione relativa all'array A (con dimA elementi) e la restituisca col return. La tabella deve essere costruita dinamicamente. E' ammessa anche una soluzione ricorsiva.
2) Il secondo problema riguarda invece una forma di pattern matching in cui sono ammessi errori come spiegato dal seguente esempio:
   siano T= "abbabab" e P="aab", rispettivamente il testo ed il pattern da cercare nel testo. Ci sono 2 match di P in T, il primo considera T[0]='a', T[3]='a' e T[4]='b', mentre il secondo match considera T[3]='a', T[5]='a' e T[6]='b'. La **distanza** di un match è la misura dei "buchi" che lascia tra i match in T dei successivi caratteri di P. Quindi nel nostro esempio, la distanza del primo match è: (3-0-1)+(4-3-1)=2, mentre quella del secondo match è: (5-3-1)+(6-5-1)=1. Si osservi che un match che non lascia buchi, cioè in T[i], T[i+1] e T[i+2], ha distanza (i+1-i-1)+(i+2-(i+1)-1)=0.

   Si chiede di scrivere una funzione che riceve T e P e le loro dimensioni come parametri e che calcola se esiste un match di P in T, e in questo caso trova il match di distanza minima restituendo appunto la sua distanza e la posizione di T in cui il match ha inizio.
   Nel nostro esempio la funzione richiesta dovrebbe calcolare: true (c'è match), 1 e 3 (in T[3] inizia il match migliore, cioè quello con distanza 1).

Parte ricorsiva: concerne le liste concatenate i cui nodi hanno il solito tipo: struct nodo{int info; nodo* next;};
Si chiede di scrivere una funzione ricorsiva con il seguente prototipo: nodo * F(nodo *& L, int l1, int l2) che deve togliere i nodi di L che hanno campo info compreso nell'intervallo l1,l2 (compresi gli estremi) e restituisce sia la lista L rimanente, sia (col return) la lista degli elementi estratti nell'ordine che essi possedevano nella lista originale.

Esempio: L= 7->3->8->2->5->0->4 , sia l1=2 e l2= 4, allora le liste che la funzione deve restituire devono essere come segue:
L deve diventare, 7->8->5->0
mentre quella restituita col return è, 3->2->4