Esame di Fondamenti di Programmazione
9-2-09 – canale AD – Sterbini
Compito A
Prima Parte

(da svolgere se non si è superato l'esonero)

Esercizio 1

Si scriva la funzione C che riceve come argomenti: una stringa **parola** ed un vettore di stringhe **dizionario** e tutti gli altri argomenti che ritenete necessari, e che cerca nel dizionario la parola che più le si avvicina, ovvero la parola che la contiene e differisce per il minimo numero di caratteri aggiunti (nel caso migliore 0 se sono uguali).
La funzione deve tornare (in due variabili passate per riferimento) l'indice della parola trovata e il numero di caratteri differenti (0 se le parole sono uguali). Se il dizionario non contiene parole simili si torni -1 -1.
Consiglio: suddividete il programma in funzioni più semplici.
Esempio: se la parola è "mamma", la parola "mammella" le è simile (la contiene ed ha 3 caratteri aggiunti)

Seconda Parte

(da svolgere obbligatoriamente)

Esercizio 2

Sia dato un tipo struct definito come segue:

```
typedef struct nodo {
  long val;
  struct nodo * figlio_sx;
  struct nodo * figlio_dx;
} Nodo;
```

Ricordiamo che un albero binario di ricerca è un albero in cui ogni nodo contiene un valore strettamente maggiore di tutti i valori contenuti nel suo sottoalbero sinistro e strettamente minore di tutti i valori contenuti nel suo sottoalbero destro.

Scrivere una funzione RICORSIVA `riempi_albero` che riceve come argomenti un puntatore a `Nodo` (e tutti gli altri argomenti che ritenete necessari). Il puntatore a `Nodo` rappresenta la radice di un albero binario di ricerca; i campi val dell'albero ricevuto sono inizializzati con il valore -1 e, al termine della chiamata alla funzione, dovranno contenere valori >0. I valori devono essere tutti gli interi da 1 a N (se l'albero contiene N nodi) e devono essere disposti in modo da soddisfare la proprietà dell'albero di ricerca.

Esercizio 3

Supponete di avere una funzione di libreria `testa_elemento` con il seguente prototipo:
`int testa_elemento (long val, int pos);`

La funzione `testa_elemento` descrive cosa va fatto su ciascun elemento della lista e restituisce:

- -1 se l'elemento con valore `val` che si trova nella posizione `pos` va eliminato dalla lista;
- 0 se l'elemento con valore `val` che si trova nella posizione `pos` va lasciato inalterato;
- un valore positivo k se l'elemento con valore `val` che si trova nella posizione `pos` va incrementato di k.

Scrivete quindi una funzione `modifica_lista` che riceve come argomenti una lista di interi (e tutti gli altri parametri che ritenete necessari) e la modifica in base ai risultati tornato da testa_elemento. Ricordate di fornire la definizione del tipo struct necessario ad implementare tale lista.

Esame di Fondamenti di Programmazione
9-2-09 – canale AD – Sterbini
Compito B
Prima Parte

(da svolgere se non si è superato l'esonero)

Esercizio 1

Si scriva la funzione C che riceve come argomenti: una matrice RxC di interi M e tutti gli altri argomenti che ritenete necessari, e che calcola quante volte nella matrice sono presenti 3 interi consecutivi, disposti orizzontalmente, verticalmente o in diagonale.
La funzione deve tornare, in due variabili passate per riferimento, le coordinate X,Y del primo numero della terna più grande trovata.

Seconda Parte

(da svolgere obbligatoriamente)

Esercizio 2

Sia dato un tipo struct definito come segue:

```
typedef struct nodo {
  long val;
  struct nodo * figlio_sx;
  struct nodo * figlio_dx;
} Nodo;
```

Ricordiamo che un albero binario di ricerca è un albero in cui ogni nodo contiene un valore strettamente maggiore di tutti i valori contenuti nel suo sottoalbero sinistro e strettamente minore di tutti i valori contenuti nel suo sottoalbero destro.

Scrivere una funzione RICORSIVA `somma_sinistro` che riceve come argomenti un puntatore a `Nodo` (e tutti gli altri argomenti che ritenete necessari). Il puntatore a `Nodo` rappresenta la radice di un albero binario di ricerca i cui valori sono tutti positivi; la funzione restituirà un valore >0 se il campo val di ogni nodo contiene un valore pari alla somma dei campi val del suo sottoalbero sinistro. Negli altri casi la funzione restituirà -1.

Esercizio 3

Supponete di avere una funzione `trova_massimo` con il seguente prototipo:
`int * trova_massimo (int * vett, int * utilizzati);`

La funzione restituisce l'indirizzo della cella con valore massimo ancora non selezionato all'interno del vettore `vett`, impostando ad 1 la cella corrispondente nel vettore `utilizzati`. (Il vettore in input avrà lunghezza ignota, conterrà interi positivi e sarà terminato dal valore -1.)

Scrivete quindi una funzione `lista_ordinata` che riceve come argomenti un vettore di interi ed il corrispondente vettore "utilizzati" (e tutti gli altri parametri che ritenete necessari) e che, chiamando più volte la funzione trova_massimo, costruisce e restituisce una lista, i cui elementi puntino agli elementi del vettore per permettere di scorrere il vettore in ordine non decrescente. Ricordate di fornire la definizione del tipo struct necessario ad implementare tale lista – implementate gli inserimenti tramite una pila.

Esame di Fondamenti di Programmazione
9-2-09 – canale AD – Sterbini
Compito C
Prima Parte

(da svolgere se non si è superato l'esonero)

Esercizio 1

Si scriva la funzione C che riceve come argomenti: un vettore di stringhe **testo** che contiene stringhe (frasi formate da parole separate da spazi) e tutti gli altri argomenti che ritenete necessari, e che stampa tutte le frasi che sono palindrome **ignorando gli spazi**.
Consiglio: suddividete il programma in funzioni più semplici.
Esempio: la frase “amo Roma” è palindroma (se ignoro lo spazio la si può leggere uguale da SX e da DX)

Seconda Parte

(da svolgere obbligatoriamente)

Esercizio 2

Sia dato un tipo struct definito come segue:

```
typedef struct nodo {
  long val;
  struct nodo * figlio_sx;
  struct nodo * figlio_dx;
} Nodo;
```

Ricordiamo che un albero binario di ricerca è un albero in cui ogni nodo contiene un valore strettamente maggiore di tutti i valori contenuti nel suo sottoalbero sinistro e strettamente minore di tutti i valori contenuti nel suo sottoalbero destro.

Scrivere una funzione RICORSIVA `bilanciato` che riceve come argomenti un puntatore a `Nodo` (e tutti gli altri argomenti che ritenete necessari). Il puntatore a `Nodo` rappresenta la radice di un albero binario di ricerca; la funzione restituirà un valore >0 se l'albero è bilanciato, -1 altrimenti. Consideriamo un albero binario bilanciato se, per ogni nodo, i nodi del sottoalbero più piccolo sono almeno la parte intera inferiore della metà dei nodi del sottoalbero più grande.

Un secondo modo per verificare se un albero binario di ricerca è bilanciato è: controllare che la profondità massima e la profondità minima di ciascun albero e sottoalbero differiscano al massimo di 1.

Esercizio 3

Scrivere una funzione `somma_e_elimina` che riceve come argomenti una lista di interi, un intero `inizio` e un intero `quanti` (e tutti gli altri argomenti che ritenete necessari). La funzione deve eliminare dalla lista (deallocando correttamente la memoria), un numero di elementi pari al valore di `quanti` (o fino al termine della lista se non ci sono abbastanza elementi), a partire da quello in posizione nella lista pari al valore di `inizio`. La funzione deve restiture la somma dei valori eliminati. Ricordate di fornire la definizione del tipo struct necessario ad implementare la lista.

Esame di Fondamenti di Programmazione
9-2-09 – canale AD – Sterbini
Compito D
Prima Parte

(da svolgere se non si è superato l'esonero)

Esercizio 1

Si scriva la funzione C che riceve come argomenti: una matrice RxC di interi M e tutti gli altri argomenti che ritenete necessari, e che ordina gli elementi di ciascuna riga della matrice con valori crescenti da sinistra a destra, ed inolte ordina tra loro le righe in ordine lessicografico crescente (ovvero in modo che una riga che ha il primo elemento minore del primo di un'altra si trovi prima, e se invece i due primi elementi sono uguali, venga prima la riga che ha il secondo elemento minore … eccetera.
Consiglio: suddividete il programma in funzioni più semplici.
Esempio di matrice disordinata e ordinata:

4	2	1	3
2	10	9	5
8	7	9	2
9	7	10	2

1	2	3	4
2	5	9	10
2	7	8	9
2	7	9	10

Seconda Parte

(da svolgere obbligatoriamente)

Esercizio 2

Sia dato un tipo struct definito come segue:

```
typedef struct nodo {
  long val;
  struct nodo * figlio_sx;
  struct nodo * figlio_dx;
} Nodo;
```

Ricordiamo che un albero binario di ricerca è un albero in cui ogni nodo contiene un valore strettamente maggiore di tutti i valori contenuti nel suo sottoalbero sinistro e strettamente minore di tutti i valori contenuti nel suo sottoalbero destro.

Scrivere una funzione RICORSIVA `exp` che riceve come argomenti un puntatore a `Nodo` (e tutti gli altri argomenti che ritenete necessari). Il puntatore a `Nodo` rappresenta la radice di un albero binario di ricerca in cui tutti i valori sono positivi; la funzione restituirà un valore >0 se, per ogni nodo, il valore del figlio sinistro è minore o uguale alla metà del valore del padre e il valore del figlio destro è maggiore o uguale al doppio del valore del padre.

Esercizio 3

Supponete di avere una funzione `trova_minimo` con il seguente prototipo:
`int * trova_minimo (int * vett, int * utilizzati);`

La funzione restituisce l'indirizzo della cella con valore minimo ancora non selezionato all'interno del vettore `vett`, impostando ad 1 la cella corrispondente nel vettore `utilizzati`. (Il vettore in input avrà lunghezza ignota, conterrà interi positivi e sarà terminato dal valore -1.)

Scrivete quindi una funzione `lista_ordinata` che riceve come argomenti un vettore di interi ed il corrispondente vettore "utilizzati" (e tutti gli altri parametri che ritenete necessari) e che, chiamando più volte la funzione trova_minimo, costruisce e restituisce una lista, i cui elementi puntino agli elementi del vettore per permettere di scorrere il vettore in ordine non decrescente. Ricordate di fornire la definizione del tipo struct necessario ad implementare tale lista – implementate gli inserimenti tramite una coda.