Laboratorio Informatica A
Problema 1

Partendo dal file lista.c si scriva una funzione che ricevute due liste di interi calcola la media degli elementi comuni tra le due liste.

Problema 2

La seguente struttura dati rappresenta un elenco di parole “proibite”, per le quali si registrano le occorrenze e dei messaggi associati. L’elenco è rappresentato come un vettore di strutture, delle quali solo le prime NumParole sono significative. Nella piattaforma, sizeof(int)= sizeof(void *) = 4

```
typedef char Parola[MAX];
typedef struct {
        Parola word;
        char * messaggio;
        int occorrenze;
} ParolaProibita;
typedef ParolaProibita ListaParole[1000];
typedef struct {
        ListaParole lista;
        int NumParole;
} ElencoParole;
ElencoParole ep, *x = &ep;
```

L’elenco è usato per monitorare la frequenza d’uso di parole sconvenienti in un forum. Quando una di tali parole viene “intercettata”, un sistema automatico visualizza il messaggio associato alla specifica parola.

Si codifichi una funzione C ...scorri(...) che, ricevuti come parametri un carattere c e un vettore di caratteri v lungo MAX, sposta “indietro” di una posizione tutti i caratteri di v tranne il primo, e riempie il “vuoto” formatosi in fondo con c. Si badi a definire correttamente l’intestazione.

Si codifichi una funzione C ...controlla(...) che riceve come parametri una parola p e un elenco di parole e, cerca la p in e, se la trova aggiorna il numero di occorrenze, stampa su stdout il messaggio associato e restituisce 1, e invece se non la trova, si limita a restituire 0.

Si codifichi un programma che legge da stdin una sequenza di caratteri a priori illimitata, derivata dal flusso di messaggi postati sui forum e nelle chat, e ignorando tutti i caratteri che non sono spazi, new line ('\n') o caratteri alfabetici, verifica se gli utenti utilizzano le parole “proibite” raccolte in un elenco, aggiornandone i dati sulle occorrenze e visualizzando i messaggi di avvertimento.

Problema 3

Considerate la seguenti strutture

```
typedef struct _simpleListNode {
   int data;
   struct * _simpleListNode next;
} simpleListNode;

simpleListNode * simpleList;

typedef struct _listOfListsNode {
   int data;
   simpleList sideList;
   struct * _listOfListsNode next;
} listOfListsNode;

listOfListsNode * listOfLists;
```

Questa struttura definisce il nodo di una lista di liste. In pratica, ogni nodo della lista contiene un dato (data) di tipo int, un puntatore al prossimo nodo (next), e un puntatore ad una lista differente (sideList). Costruire una funzione che accetta in input una lista di liste (come definito sopra) e restituisce una lista semplice che contiene l’appiattimento della lista in input. Il prototipo della funzione da creare è il seguente: simpleListNode flatten(listOfListsNode listOfLists)

Problema 4

Si consideri una matrice (o griglia) di caratteri di dimensione NxN, definita come segue:

typedef char griglia[N][N];

(a) Si codifichi in C una funzione char ennuplaVert(griglia g) che, se esiste una colonna in g costituita da un allineamento verticale di N caratteri tutti uguali, restituisce il carattere presente su tale colonna, altrimenti restituisce il carattere '\0'. Si trascuri il caso in cui più colonne godano della proprietà da verificare, considerando il primo allineamento rilevato (ad es., il più a sinistra).

(b) Si codifichi una funzione char ennuplaOriz(griglia g) che effettui analoga verifica per un allineamento orizzontale.

(c) Si codifichi in C una funzione char ennuplaDiag(griglia g) che verifica in modo analogo alle precedenti se vi è un allineamento su una delle due diagonali principali.

(d) Nel popolarissimo gioco del tic tac toe (tris in italiano) i giocatori tentano di disporre 3 simboli uguali su una griglia 3x3 apponendovi a turno delle 'X' (il simbolo del 1° giocatore) e delle 'O' (simbolo del 2°). Nella griglia a sotto la partita è terminata in parità (come sempre succede, se nessun giocatore commette errori). Vogliamo generalizzare il gioco al caso NxN, usando le definizioni dei punti precedenti. Si codifichi in C una funzione int esito(griglia g) che restituisce 0 se la partita rappresentata da g non è finita, 1 se ha vinto il "giocatore X", 2 se ha vinto il "giocatore O", 3 se è un pareggio.

(e) Si consideri la seguente definizione di una partita rappresentata come lista di mosse, in cui ogni mossa è una coppia di interi compresi tra 0 e N-1 che indicano la riga e la colonna in cui i giocatori appongono i simboli. La prima mossa è del giocatore X, la seconda del giocatore O, … . Si assuma per semplicità che le mosse di una partita siano sempre valide (coordinate corrette, simboli non sovrapposti, …).

```
typedef struct mo { int  r,c;
    struct mo * next; } Mossa;
typedef Mossa * Partita;
```

Si codifichi in C una funzione ...gioca(...) che riceve come parametro una Partita e ne indica l'esito finale secondo le convenzioni del punto precedente (simulandone opportunamente l'esecuzione, applicando le mosse ad una griglia inizialmente vuota).

O	X	X
X	O	O
O	X	x