

ESAME INTEGRATO PROG I +PROG II
19 Settembre 2007 – Canale II, Lettere E—O.

Nome

Cognome

Matricola

ISTRUZIONI

- 1 Chi sostiene **solo Prog 1** deve svolgere i Problemi **A,B,C**. **Tempo concesso: 2h 15 min.**
- 2 Chi sostiene **solo Prog2** deve svolgere i problemi **D,E,F,G**. **Tempo concesso: 2h30min.**
- 3 Chi sostiene **la prova integrata Prog1+Prog2** deve svolgere **due** problemi a scelta tra **A,B,C** e **tre** problemi a scelta tra **D,E,F,G**. **Tempo concesso: 3 h:**
- 4 **Non Consegno**

SCelta EFFETTUAATA

Ho scelto l'opzione _____

ESAME ORALE e VERBALIZZAZIONE:
Venerdi 21 Settembre ore 9:00 Stanza Docente (N. Galesi)

PROBLEMA A

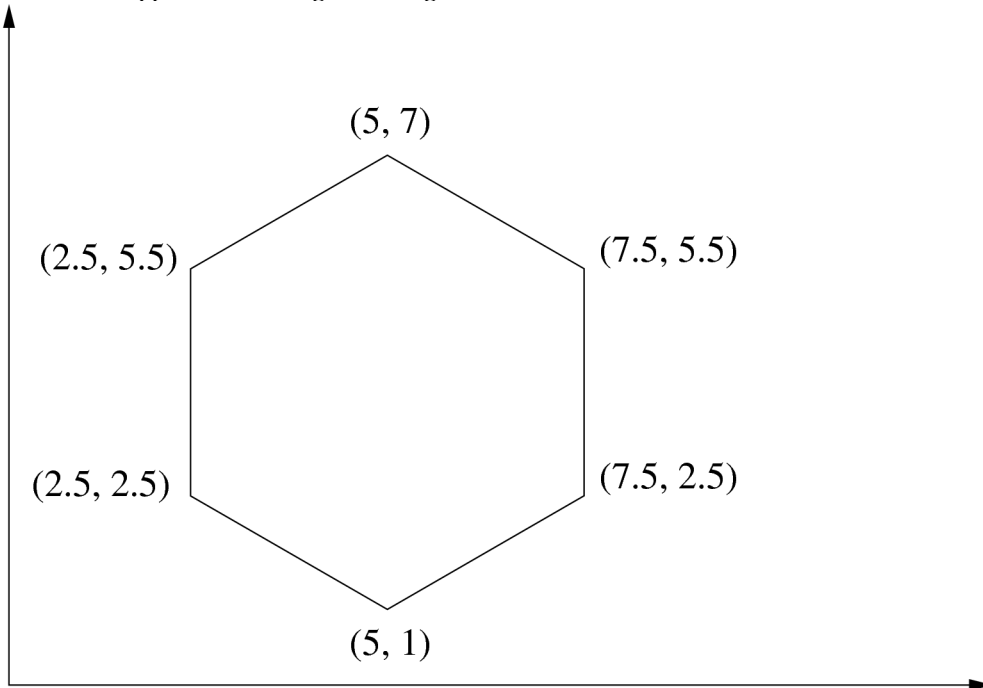
Sia A una matrice di numeri interi di N righe ed M colonne, ordinata in maniera crescente sia per righe che per colonne. Si disegni una funzione che implementi un **algoritmo iterativo** (con una sola iterazione) che dati in input A , le sue dimensioni e un intero x , dica se x è in A e in caso positivo in quale posizione. Si studi l'iterazione specificando: (1) variabili coinvolte e loro significato; (2) loro inizializzazione nel ciclo, (3) corpo del ciclo (4) condizione di terminazione. Si scriva infine l'algoritmo in C specificando pre e post condizione della funzione.

PROBLEMA B

Disegnare ed implementare una **funzione primo** che, in input un numero naturale $n \geq 1$, dica se n è primo o meno. La funzione primo deve essere ottenuta da una funzione `no_divisori(int n, int d)` che decide se n ha divisori minori o uguali di d e maggiori o uguali 2. Si ricordi che un naturale n non può avere divisori più grandi di \sqrt{n} . Si disegni ricorsivamente la funzione `no_divisori` descrivendo dettagliatamente: **caso base**, **caso ricorsivo**, **terminazione della ricorsione**, **pre e post-condizione della funzione**.

PROBLEMA C

Un poligono è definito come la lista dei suoi vertici, rappresentati tramite coordinate cartesiane. Ad esempio, se si vuole rappresentare il seguente esagono:



lo si farà con una lista del tipo: (5, 1) --> (2.5, 2.5) --> (2.5, 5.5) --> (5, 7) --> (7.5, 5.5) --> (7.5, 2.5)
Scrivere:

- Due strutture dati `punto` e `poligono` che definiscano, rispettivamente, un punto nel piano ed un poligono definito come sopra;
- Una funzione `is_regular` che prende come parametri un poligono p e ritorna in output 1 se p è regolare e 0 altrimenti. A tal proposito, si supponga che esista un file `funzioni_sul_piano.h` che definisce ed implementa le seguenti funzioni:

```
double distanza(punto A, punto B);  
/* Pre: A e B sono punti nel piano  
   Post: distanza(A,B)=lunghezza del segmento AB */  
double angolo(punto B, punto A, punto C);  
/* Pre: A, B e C sono punti nel piano  
   Post: angolo(A,B,C)= la misura dell'angolo BÂC */
```

Si ricorda che un poligono è regolare se e solo tutti i suoi lati e tutti i suoi angoli sono uguali.

PROBLEMA D

Si supponga di dover memorizzare un dizionario D contenente M chiavi in una **tabella hash** H implementata mediante liste di trabocco. Si supponga che la tabella hash abbia dimensione MAX_TAB e che le liste di trabocco possono contenere al massimo MAX_LST elementi. Si considerino le seguenti dichiarazioni che definiscono H e il prototipo delle funzione **hash**:

```
typedef struct elem_lst{
    chiave k;
    struct elem_lst *next;
}elem_lst

typedef elem_lst *lst_trbc;

typedef struct elem_tab{
    lst_trbc lst;      /* ptr alla lista di trabocco */
    int cnt;          /* num. di elementi correnti nella lista di trabocco<=
                      MAX_LST*/
} elem_tab;

elem_tab H[MAX_TAB];

int hash(chiave K)
/* Post: hash(k) restituisce la posizione i in H dove memorizzare i dati
   relativi alla chiave K */
```

Sapendo che sempre si avrà che $M < \text{MAX_TAB} * \text{MAX_LST}$, si proponga un criterio per memorizzare il dizionario usando H, si scriva una funzione **mod_hash** che usi **hash** e implementi il criterio scelto, e infine si scriva una funzione che consenta la ricerca di una chiave nella tabella.

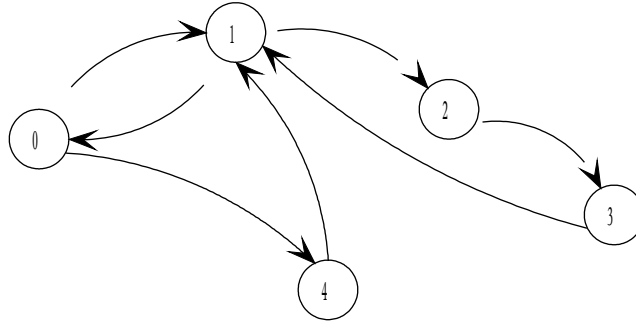
PROBLEMA E

Dichiarare e definire in linguaggio C i dati e le funzioni necessarie per implementare la visita per livello di un albero binario.

PROBLEMA F

Scrivere una funzione che, data una lista di interi $[k_1, k_2, \dots, k_n]$, restituisca un grafo rappresentato mediante liste di adiacenza, tale che i nodi siano gli interi nella lista e gli archi i collegamenti tra gli interi k_j e k_{j+1} con $j = 1, \dots, n-1$.

Ad esempio, data la lista $[0, 1, 0, 4, 1, 2, 3, 1]$, la funzione creerà un grafo con 5 nodi (0, 1, 2, 3 e 4) e con gli archi $(0, 1)$, $(1, 0)$, $(0, 4)$, $(4, 1)$, $(1, 2)$, $(2, 3)$ e $(3, 1)$, ovvero il grafo seguente:



PROBLEMA G

Una sequenza di n interi si dice **massimamente crescente** se il valore di ogni coordinata (esclusa la prima) è strettamente maggiore del massimo delle coordinate anteriori. Si scriva un algoritmo che generi tutte le sequenze di n elementi nell'insieme $\{1, \dots, k\}$ massimamente crescenti. Ad esempio se $k=5$ ed $n=3$ il programma deve stampare le sequenze:

(1,2,3),(1,2,4),(1,2,5),(1,3,4),(1,3,5),(1,4,5),(2,3,4),(2,3,5),(2,4,5),(3,4,5).