Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compito del 2 Settembre 2008

[Esercizio 1 - punti 6]
Si scriva una procedura int *CalcSomme(int **A, int n, int k), che, data una matrice A $n \times n$ ad elementi interi, ed un intero positivo $k \leq n$, restituisca un vettore di interi di lunghezza $(n-k+1)^2$ contenente le somme degli elementi di tutte le sottomatrici principali di ordine k contenute nella matrice A.

[Esercizio 2 - punti 9]
Sia data una struttura typedef struct elemento {char *stringa; struct elemento *next} listaS; che rappresenta un elemento di una lista di stringhe. Si scriva una funzione listaS *compattaCorte (listaS *lis, int x) che, dato il puntatore lis ad una lista di stringhe, concateni tra loro stringhe consecutive di lis in modo da ottenere solo stringhe di lunghezza almeno x. In altre parole, quando si incontra una stringa di lunghezza minore di x essa deve essere concatenata alla(e) stringa(he) seguente(i) fino a quando non si ottiene una stringa di lunghezza almeno x. Le stringhe della lista risultante avranno quindi tutte lunghezza $\geq x$, eccetto eventualmente l'ultima. Ad esempio, se lis contiene le stringhe {so, p, oroe, poi, mihon, giob, w, ab} ed $x = 4$, la lista risultante dovrà contenere le stringhe {soporoe, poimihon, giob, wab}. La procedura deve restituire un puntatore alla lista così modificata. Eventuale memoria non più utilizzata deve essere correttamente deallocata. È ammesso l'utilizzo della funzione di libreria strlen.

[Esercizio 3 - punti 9]
Elena ha guadagnato 400 euro dando ripetizioni e vuole utilizzarli per acquistare dei libri. Elena scopre che i prezzi migliori sono quelli offerti da due librerie online. Rappresenta quindi un libro con la struttura typedef struct {int grad; int p1; int p2} libro; dove grad indica il gradimento del libro, p1 il prezzo del libro presso la prima libreria, p2 il prezzo presso la seconda libreria. Ci sono inoltre le spese di spedizione: per la prima libreria sono 5 euro fissi più 2 euro per volume, mentre per la seconda libreria sono 11 euro fissi più 1 euro per volume. Scrivere una procedura void amazon(libro a[], int n) che dato un array di n libri, stampi l'elenco dei libri da acquistare presso ciascuna libreria al fine di massimizzare il gradimento totale (definito semplicemente come la somma dei gradimenti dei singoli libri acquistati).

[Esercizio 4 - punti 4+4]
Data una stringa $\beta \in \{0,1\}^*$ si indica con $(\beta)_2$ il valore di β interpretata come numero binario.

1. Si consideri il linguaggio $L_1 \subseteq \{0,1,a\}^*$ composto dalle stringhe della forma βa^n, tali che $\beta \in \{0,1\}^*$, $|\beta| > 0$, $n > 0$, e $(\beta)_2 > n$. Un esempio di stringa appartenente al linguaggio è $110aaaa$ in quanto $(110)_2 = 6$. Si costruisca un automa a stati finiti che riconosce L_1 oppure si dimostri che un tale automa non può esistere.
2. Si ripeta l'esercizio per il linguaggio $L_2 \subseteq \{0,1,a\}^*$ composto dalle stringhe della forma βa^n, tali che $\beta \in \{0,1\}^*$, $|\beta| > 0$, $n > 0$, e $(\beta)_2 + n^2 \equiv 3 \bmod 7$. Un esempio di stringa appartenente al linguaggio è $1000aaa$ in quanto $(1000)_2 = 8$ e $8 + 3^2 \equiv 3 \bmod 7$.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compito del 27 Giugno 2008

[Esercizio 1 - punti 8]
Scrivere una procedura `int DoppiaSequenza(int n)` che, dato un intero $n > 0$, simuli una successione randomizzata di lanci di moneta e termini quando sono state generate una sequenza $(TC)^n$ e una sequenza $(CT)^n$ (notare che entrambe le sequenze hanno lunghezza $2n$). Le due sequenze possono essere generate in qualsiasi ordine, ma devono essere disgiunte (cioè non avere elementi in comune). La procedura deve restituire il numero di lanci che è stato necessario effettuare affinché la condizione di terminazione sia verificata.

[Esercizio 2 - punti 8]
Sia data una struttura `typedef struct elemento {int val; struct elemento *next;} lista;` che rappresenta una lista *circolare ordinata* (in senso crescente) di interi. (Si ricorda che in una lista circolare, il campo `next` dell'ultimo elemento della lista punta al primo elemento della lista). Si scriva una procedura `lista *RimuoviIntervallo(lista *lis, int a, int b)` che, dato un puntatore `lis` ad un *qualsiasi elemento* (non necessariamente il più piccolo) di una lista circolare ordinata, e due interi a e b con $a \leq b$, cancelli da `lis` tutti gli elementi di valore compreso fra a e b (estremi compresi), e restituisca il puntatore ad un *qualsiasi* elemento della lista così ottenuta.

[Esercizio 3 - punti 8]
Sia data una struttura `typedef struct nodo {char car; struct nodo *sx; struct nodo *dx;} albero;` che rappresenta un nodo di un albero binario contenente un singolo carattere. Data una stringa s di lunghezza k si dice che una catena di nodi $n_0, n_1, \ldots, n_{k-1}$ contiene s se per $i = 0, \ldots, k-1$ il nodo n_i contiene il carattere $s[i]$ e per $i = 1, \ldots, k-1$ il nodo n_i è un figlio del nodo n_{i-1}. Si scriva una procedura `int conta(albero *a, char *s)` che dato il puntatore `a` alla radice di un albero binario non vuoto e una stringa non vuota `s`, restituisca il numero di catene dell'albero contenenti `s`.

[Esercizio 4 - punti 4+4]
Dovete preparare il DVD del matrimonio del vostro miglior amico. Avete un filmato di n fotogrammi e dovete suddividerlo in un certo numero di capitoli. Per automatizzare la procedura avete calcolato un array di `unsigned char` s tale che $s[i]$ contiene una stima della "similarità" fra il fotogramma i e il fotogramma $i+1$: più grande è il valore $s[i]$ maggiore è la similarità fra i fotogrammi. I punti di inizio capitolo devono essere messi preferibilmente dove ci sono i cambi di scena; per questo motivo un punto di inizio capitolo messo immediatamente dopo il fotogramma i viene penalizzato di una quantità $s[i]$. Scrivere una procedura `void chapters(unsigned char *s, int n, int k)` che dato l'array s e il parametro k stampi l'elenco dei punti di inizio capitolo che minimizza la somma delle penalizzazioni rispettando il vincolo che ogni capitolo non deve contenere più di k fotogrammi. In termini matematici si deve quindi determinare la sequenza $c_1, c_2, \ldots, c_h$ che minimizza $\sum_{i=1}^{h} s[c_i]$ rispettando i vincoli: $0 \leq c_1 < k$, $c_h \geq n-k-1$, e $1 \leq c_i - c_{i-1} \leq k$ per $i = 2, \ldots, h$. I punti extra saranno assegnati a chi propone una soluzione di costo polinomiale rispetto a n.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compito del 3 Giugno 2008

[Esercizio 1 - punti 6]
Una matrice quadrata A $n \times n$ ad elementi 0/1, si dice avere una cornice di ordine k con $1 \leq k \leq n$ se i suoi elementi a_{ij} sono tali che

$$i \in [0, k-1] \cup [n-k, n-1] \Rightarrow a_{ij} = 1 \quad \text{e} \quad j \in [0, k-1] \cup [n-k, n-1] \Rightarrow a_{ij} = 1.$$

Si scriva una funzione int `CalcolaCorniceMax(int **A, int n)` che data una matrice $n \times n$ `A`, restituisca il massimo valore di k tale che `A` contiene una cornice di ordine k.

[Esercizio 2 - punti 9]
Si considerino le strutture `typedef struct lstr {char *str; struct lstr *next;} listastr;` che rappresenta una lista di stringhe, e `typedef struct lcar {char car; struct lcar *next; listastr *first;} listacar;` che rappresenta una lista di caratteri con un campo addizionale `first` utilizzato nel seguito. Si scriva una funzione `listacar *CreaIndice(listastr *Ls)` che data una lista *ordinata* di stringhe `Ls` costruisca e restituisca una lista *ordinata* di caratteri `Lc` tale che 1) `Lc` contiene esattamente i caratteri che sono iniziali di stringhe in `Ls`, 2) per ogni carattere c che appare in `Lc` il campo `first` dell'elemento c punta al primo elemento di `Ls` contenente una stringa che inizia per c. Ad esempio se `Ls` contiene le stringhe: agnello, albatros, cobra, elefante, tigre, topo, la lista `Lc` deve contenere i caratteri: `a`, `c`, `e`, `t` e i rispettivi campi `first` devono puntare agli elementi di `Ls` contenenti le stringhe: agnello, cobra, elefante, tigre.

[Esercizio 3 - punti 9]
Ernesto deve percorrere M chilometri in motocicletta. Prima di partire si segna la posizione delle n stazioni di servizio $s_0, s_1, \ldots, s_{n-1}$ presenti lungo il percorso. Tali posizioni sono identificate dalle distanze (in chilometri) $d_0, d_1, \ldots d_{n-1}$ dove d_0 è la distanza dal punto di partenza alla stazione s_0, e per $i = 1, \ldots, n-1$ d_i è la distanza fra le stazioni s_{i-1} e s_i. Inoltre, per $i = 0, \ldots, n-1$, p_i indica il prezzo (al litro) praticato nella stazione s_i. La motocicletta consuma 0.05 litri per chilometro e ha un serbatoio di 30 litri inizialmente pieno. Ernesto decide di riempire totalmente il serbatoio ogni volta che si ferma in una stazione di servizio. Scrivere una procedura `void pianifica(double M, int n, double d[], double p[])` che dati gli array delle distanze `d[]` e dei prezzi `p[]` stampi l'elenco delle stazioni di servizio in cui si deve fermare Ernesto per minimizzare la spesa per il carburante (si ignori il valore del carburante che rimane nel serbatoio al termine del viaggio).

[Esercizio 4 - punti 4+4]

1. Si costruisca un automa a stati finiti che riconosca il linguaggio composto dalle stringhe binarie che a) iniziano per **1**, b) interpretate come numero binario (nella maniera usuale) hanno un valore $\equiv 2 \bmod 5$, c) contengono almeno due **1** consecutivi. Ad esempio, la stringa **101111** appartiene al linguaggio in quanto corrisponde al valore decimale $2^5 + 2^3 + 2^2 + 2^1 + 2^0 = 47$.
2. Si dimostri che non esiste un automa a stati finiti che riconosce il linguaggio composto dalle stringhe binarie che a) iniziano per **1**, b) interpretate come numero binario (nella maniera usuale) hanno un valore $\equiv 2 \bmod 5$, c) contengono un numero di **1** maggiore o uguale al numero di **0**. Esempio di stringhe appartenenti a questo linguaggio sono **111**, **1100**, **10101**, in quanto corrispondono rispettivamente ai valori decimali 7, 12, e 22.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compitino del 27 Maggio 2008

[Esercizio 1 - punti 6]
Scrivere una procedura `int **pascal(int n)` che dato un intero $n > 0$ restituisca un array di `int *` contenente le prime $n+1$ righe del triangolo di Pascal. In altre parole se `p=pascal(n)`, per $k = 0, 1, \ldots, n$ e $i = 0, \ldots, k$, `p[k][i]` deve contenere il coefficiente binomiale $\binom{k}{i}$. Se la memoria non è sufficiente per allocare l'intero triangolo di Pascal, la procedura deve restituire `NULL` e deallocare tutta la memoria già utilizzata.

[Esercizio 2 - punti 7]
Sia data una struttura `typedef struct nodo {int val; struct nodo *sx; struct nodo *dx;} albero;` che rappresenta un nodo di un albero binario a valori interi. Diciamo che un nodo è *dominante* se non è una foglia e contiene un valore maggiore della somma dei valori dei suoi figli. Si scriva una procedura `int ContaDominanti(albero *a)` che, dato il puntatore `a` alla radice di un albero binario, restituisca il numero di nodi dominanti contenuti nell'albero di radice `a`.

[Esercizio 3 - punti 8]
Sia data una struttura typedef struct elemento {int val; struct elemento *next} lista; che rappresenta un elemento di una lista di interi. Si scriva una procedura lista *SeekAndDestroy(lista *lis, int k) che data una lista lis di interi positivi e un intero positivo k, cerca all'interno di lis la prima sequenza di elementi consecutivi la cui somma sia esattamente k e elimina tali elementi dalla lista. La funzione deve restituire un puntatore al primo elemento della lista così modificata. La valutazione terrà conto della semplicità ed efficienza della soluzione proposta. [Nota: deve essere eliminata solo la prima sequenza di elementi con somma k; una tale sequenza potrebbe non esistere: in tal caso la lista non deve essere modificata.]

[Esercizio 4 - punti 7+5]
Dovete aiutare vostra zia a portare l'elettricità in giardino utilizzando delle prolunghe. Ogni prolunga ha un connettore maschio e uno femmina ma i connettori possono essere di tre tipi diversi che indichiamo con 0, 1, e 2. Per rappresentare una prolunga si usa il tipo prolunga così definito: typedef struct {int lung; char cf; char cm;} prolunga. Se p è una prolunga p.lung è la sua lunghezza in metri, p.cf $\in \{0, 1, 2\}$ è il tipo del suo connettore femmina, p.cm $\in \{0, 1, 2\}$ è il tipo del suo connettore maschio. La prolunga q può essere attaccata alla prolunga p se e solo se p.cf==q.cm.

1. Scrivere una procedura int maxlung(prolunga a[ ], int n) che dato un array di n prolunghe a[0], a[1], ..., a[n-1] trova la massima lunghezza di una catena di prolunghe a[i_0], a[i_1], ..., a[i_k] compatibili, cioè tale che per $h = 0, 1, \ldots, k-1$ la prolunga a[i_{h+1}] può essere attaccata a a[i_h] (quindi a[i_h].cf== a[i_{h+1}].cm). Insieme al codice si deve fornire una descrizione sintetica della strategia di risoluzione utilizzata.
2. Mentre state programmando sopraggiunge la vostra cuginetta di dieci anni che, osservando il vostro codice, esclama: "Chissa come si complica il problema se si considerano solo le catene che soddisfano al vincolo $i_0 < i_1 < \cdots < i_k$!" (in altre parole le prolunghe devono apparire nelle catene nello stesso ordine con cui appaiono nel vettore a[ ]). Mostrate a vostra cugina che si sbaglia scrivendo una procedura che, utilizzando la programmazione dinamica, risolve in tempo polinomiale il nuovo problema.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compitino del 16 Aprile 2008

[Esercizio 1 - punti 7]
Scrivere una procedura `int SequenzeLanci()` che genera una successione pseudo-casuale di lanci di dado e termina non appena viene generata una sequenza di valori del tipo:

$$i,\ i+1,\ i+2,\ i+1,\ i$$

per un intero i con $1 \leq i \leq 4$ (questi cinque valori devono essere generati consecutivamente). La procedura deve restituire il numero totale di lanci che è stato necessario effettuare per generare la sequenza del tipo richiesto.

[Esercizio 2 - punti 7]
Scrivere una procedura `int ContaPermutazioni(int a[100][100])` che data una matrice di interi 100×100 restituisce il numero di righe che contengono una permutazione degli interi da 1 a 100. In altre parole si deve contare quante righe della matrice contengono tutti gli interi da 1 a 100 (in tal caso, dato che ogni riga ha 100 elementi, ogni intero tra 1 e 100 apparirà esattamente una volta all'interno della riga). Non è ammesso modificare in nessun modo la matrice `a[][]`.

[Esercizio 3 - punti 8]
Scrivere una procedura `int sottosuccessione(int a[], int n, int b[], int m)` che dati due array di interi $a_0, a_1, \ldots, a_{n-1}$ e $b_0, b_1, \ldots, b_{m-1}$ restituisce 1 se esistono degli indici $k_0 < k_1 < \cdots < k_{m-1}$ tali che

$$0 \leq k_0 < k_1 < \cdots < k_{m-1} \leq n-1$$

e $b_i = a_{k_i}$ per $i = 0, \ldots m-1$. La procedura deve restituire 0 se tale sequenza di indici non esiste. [Suggerimento: esiste una soluzione ricorsiva molto semplice, ma ve la potete cavare anche senza ricorsione].

[Esercizio 4 - punti 5+5]
Si consideri il linguaggio $L \subseteq \{a,b,c\}^*$ formato dalle stringhe α della forma $\alpha = (ab)^i(bc)^j(ca)^k$ con $i > 0$, $j > 0$, $k > 0$ interi positivi che soddisfano alle condizioni:

$$i + j \equiv 4 \pmod 9 \qquad \text{e} \qquad i + j + k \equiv 1 \pmod 7.$$

(Un esempio di stringa appartenente al linguaggio è *abbcbcbccacacaca* che corrisponde a $i = 1$, $j = 3$, $k = 4$). Si dimostri che esiste un automa a stati finiti deterministico che riconosce L oppure si dimostri che tale automa non può esistere. Si ripeta poi l'esercizio aggiungendo alle condizioni precedenti il vincolo che il numero di a nella stringa sia strettamente maggiore del numero di b.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compito del 22 Febbraio 2008

[**Esercizio 1 - punti 8**]
21 è un gioco di carte per due giocatori che utilizza un mazzo di carte il cui valore varia da 1 a 10. Il valore di una mano è la somma dei valori delle carte e l'obiettivo è avvicinarsi il più possibile al valore 21 senza superarlo. In ogni momento un giocatore può chiedere una nuova carta o fermarsi. Se un giocatore supera il valore 21 ottiene 0 punti, altrimenti ottiene un punteggio pari al valore della sua mano (che sarà quindi ≤ 21). Vince la partita il giocatore con il punteggio più alto; in caso di punteggi uguali non vince nessuno. Dato un intero x, con $0 < x \leq 20$, la strategia di tipo x consiste nel chiedere una nuova carta fino a quando il valore della mano è $\geq x$. Si scriva una procedura `int StrategiaMigliore(int` x_1`, int` x_2`, int` N`)` che, dati due interi *distinti* x_1, x_2 tali che $0 < x_1, x_2 \leq 20$, simuli N partite di **21**, e restituisca 1 se la strategia di tipo x_1 è risultata migliore (cioè, se ha vinto piú partite), 2 se la strategia di tipo x_2 è risultata migliore, e 0 nel caso nessuna delle due strategie ha prevalso sull'altra. Si assuma di avere a disposizione una procedura `int EstraiCarta()` che estrae una carta dal mazzo e ne restituisce il valore (compreso fra 1 e 10).

[**Esercizio 2 - punti 8**]
Sia data una struttura `typedef struct nodo{int v; struct nodo *left; struct nodo *right} albero;` che rappresenta un nodo di un albero binario di valori interi. Definiamo *fattore di sbilanciamento* dell'albero come la differenza fra la profondità massima e minima delle foglie dell'albero (si ricorda che la profondità di un nodo dell'albero è la distanza fra il nodo stesso e la radice). Si scriva una procedura `int CalcolaSbilanciamento(albero *a)` che, dato il puntatore `a` alla radice di un albero binario, restituisca il fattore di sbilanciamento dell'albero di radice `a`.

[**Esercizio 3 - punti 4+4**]
Dato un insieme di $n > 1$ città $c_0, \ldots, c_{n-1}$ sia A la matrice di **0** e **1** tale che $A[i][j] = \mathbf{1}$ se e solo se esiste un treno diretto fra le città c_i e c_j (si può supporre che A sia simmetrica e gli elementi diagonali siano **1**). Si scriva una procedura `int max2cambi(int **A, int n, int k, int h)` che restituisca 1 se è possibile andare da c_k a c_h cambiando al massimo 2 volte (cioè prendendo al massimo tre treni distinti) e restituisca 0 altrimenti. Scrivere poi una procedura `int collegate(int **A, int n, int k, int h)` che restituisca 1 se è possibile andare da c_k a c_h in treno (indipendentemente dal numero dei cambi), e restituisca 0 altrimenti. [Suggerimento: se si può andare in treno da c_k a c_h è possibile farlo cambiando al massimo $n-2$ volte.]

[**Esercizio 4 - punti 4+4**]
Si consideri il linguaggio $L \subseteq \{a,b\}^*$ formato dalle stringhe α della forma $\alpha = a^i b^j a^k$ con $i > 0$, $j > 99$, $k > 0$ interi positivi che soddisfano alle condizioni:

$$i + j \equiv 3 \pmod 7 \qquad \text{e} \qquad j + k \equiv 1 \pmod 8.$$

Si dimostri che esiste un automa a stati finiti deterministico che riconosce L oppure si dimostri che tale automa non può esistere. Si ripeta poi l'esercizio aggiungendo alle condizioni precedenti il vincolo $i^2 + k^2 > j^2$.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compito del 7 Gennaio 2008

[**Esercizio 1 - punti 6**]
Si scriva una procedura `int MaxSeq(int **a, int n)`, che, data una matrice quadrata `a` di ordine $n > 0$ ad elementi interi, restituisca la lunghezza della più lunga sottosequenza ordinata (in senso crescente) di elementi presente nelle righe o nelle colonne di `a`. La valutazione terrà conto della semplicità ed efficienza della soluzione proposta.

[**Esercizio 2 - punti 9**]
Sia data la struttura `typedef struct elemento {int lung; char *stringa; struct elemento *next} lista;`, che rappresenta un elemento di una lista di stringhe, dove il campo intero `lung` rappresenta la lunghezza della stringa (escluso il carattere di terminazione). Si scriva una funzione `lista *trasponiNonOrdinato(lista *lis);` che, dato il puntatore `lis` al primo elemento della lista, restituisca una nuova lista dello stesso tipo, dove la i-esima stringa della nuova lista è ottenuta concatenando l'elemento i-esimo delle stringhe in `lis`, quando presente. Ad esempio, se le stringhe puntate da `lis` sono {`solam, proroe, poi, mihon, ab`}, la lista risultante dovrà contenere le stringhe {`sppma, oroib, loih, aro, mon, e`}, e le rispettive lunghezze (escluso il carattere di terminazione).

[**Esercizio 3 - punti 9**]
Davide e Damiano decidono di dividersi la loro collezione di carte di Magic™. La collezione consiste di n carte; per $i = 0, \ldots, n-1$ la carta i-esima è caratterizzata da un valore di attacco a_i e un valore di difesa d_i, entrambi interi non negativi. Supponendo n pari, Davide e Damiano vogliono costruire due mazzi di esattamente $n/2$ carte tali che la somma dei valori di attacco del primo mazzo sia il più vicino possibile alla somma dei valori di attacco del secondo mazzo, e analogamente per i valori di difesa. Più precisamente, si vuole trovare due sottoinsiemi D_1, D_2 di $\{0, 1, \ldots, n-1\}$ tali che $D_1 \cap D_2 = \emptyset$, $|D_1| = |D_2| = n/2$, e tali che la quantità

$$\left|\left(\sum_{i\in D_1} a_i\right) - \left(\sum_{i\in D_2} a_i\right)\right| \quad + \quad \left|\left(\sum_{i\in D_1} d_i\right) - \left(\sum_{i\in D_2} d_i\right)\right|$$

sia minima possibile. Scrivere una procedura `void DividiMagic(int n, int *a, int *d)` che dato il numero n di carte e gli array `a[]`, `d[]` contenenti i valori di attacco e difesa di ogni carta, stampi una coppia di sottoinsiemi D_1, D_2 che soddisfi ai requisiti richiesti. Insieme al codice si deve fornire una descrizione sintetica della strategia di risoluzione utilizzata. La valutazione terrà conto della semplicità ed efficienza della soluzione proposta.

[**Esercizio 4 - punti 4+4**]
Si consideri il linguaggio $L \subseteq \{a, b\}^*$ formato dalle stringhe α della forma $\alpha = a^i b^j a^k$ con $i > 0$, $j > 0$, $k > 0$ interi positivi che soddisfano alle condizioni:

$$i + k \equiv 5 \pmod 7 \qquad \text{e} \qquad j = n^3 \text{ per un qualche intero } n.$$

(in altre parole j deve essere un cubo perfetto). Si dimostri che esiste un automa a stati finiti deterministico che riconosce L oppure si dimostri che tale automa non può esistere. Si ripeta poi l'esercizio aggiungendo alle condizioni precedenti il vincolo che la stringa α non contenga la sottostringa $b^9 a$.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compito del 19 Settembre 2007

[**Esercizio 1 - punti 7**]
Scrivere una procedura `int simula_dado()` che simula una successione pseudo-casuale di lanci di dado restituendo il minimo numero di lanci necessario affinchè entrambe le seguenti condizioni siano verificate: 1) la sequenza di lanci contiene tre lanci consecutivi con esito 1-3-5 (in questo ordine); 2) la sequenza di lanci contiene tre lanci consecutivi con esito 2-4-6 (in questo ordine). La valutazione terrà conto della semplicità ed efficienza della soluzione proposta.

[**Esercizio 2 - punti 8**]
Sia data la struttura `typedef struct elemento {int valore; struct elemento *sx; struct elemento *dx;} albero;` che rappresenta un nodo di un albero binario di ricerca i cui nodi contengono valori interi. Si scriva una funzione `albero *creaAlbero(int *V, int n, int h)` che, dato un array V di n interi, crei un albero binario di ricerca inserendo in successione gli elementi $V_0, V_1, \ldots,$ interrompendosi quando una delle seguenti condizioni risulta verificata: 1) sono stati inseriti n interi, 2) l'albero ha raggiunto l'altezza h, 3) la memoria disponibile si è esaurita. La funzione deve restituire un puntatore alla radice dell'albero binario di ricerca ottenuto. Per semplicità, si assuma che tutti gli elementi dell'array V siano distinti.

[**Esercizio 3 - punti 8**]
All'interno di un magazzino ci sono n pacchi. Ogni pacco è caratterizzato da peso, altezza, e da un limite massimo di peso che il pacco può sostenere sopra di sé (pesi, altezze e limiti sono interi positivi). Per rappresentare queste caratteristiche si utilizzi la struttura: typedef struct {int peso; int altezza; int limite;} pacco;. Scrivere una funzione int torredipisa(int n, pacco *p) che, dato un array di n pacchi, calcoli qual è l'altezza massima di una pila di pacchi che può essere formata con essi rispettando il vincolo che nessun pacco abbia sopra di sé un peso superiore al limite consentito. Insieme al codice si deve fornire una descrizione sintetica della strategia di risoluzione utilizzata. La valutazione terrà conto della semplicità ed efficienza della soluzione proposta.

[**Esercizio 4 - punti 5+4**]
Si consideri il linguaggio $L \subseteq \{a, b, c\}^*$ formato dalle stringhe della forma $a^n b^m c^p$ con m, n, p interi positivi e tali che esiste k, $0 \leq k \leq n$, per cui si ha

$$k + m \equiv 1 \pmod 2 \qquad \text{e} \qquad (n - k) + p \equiv 2 \pmod 3.$$

Si dimostri che esiste un automa a stati finiti deterministico che riconosce L oppure si dimostri che tale automa non può esistere. Si ripeta poi l'esercizio aggiungendo alle condizioni precedenti il vincolo che l'intero k deve soddisfare anche alla condizione $k + m > n - k + p$.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compito del 5 Luglio 2007

[Esercizio 1 - punti 7]
Scrivere una procedura `int mat01(char **a, int n)` che data una matrice di caratteri di dimensione $n \times n$ contenente soltanto valori 0 e 1, restituisca la dimensione della più grande sottomatrice con la proprietà che esattamente la metà dei suoi elementi sono uguali a 0. La valutazione terrà conto della semplicità ed efficienza della soluzione proposta.

[Esercizio 2 - punti 8]
Sia data una struttura `typedef struct elemento {int val; struct elemento *next} lista;` che rappresenta un elemento di una lista *circolare* di interi. In una lista circolare, il campo `next` dell'ultimo elemento invece di contenere `NULL` contiene un puntatore al primo elemento della lista. Si scriva una funzione `lista *RiOrdina (lista *l)` che, dato un puntatore `l` al primo elemento di una lista circolare ordinata in senso crescente, trova, se esiste, l'unico elemento della lista che non rispetta l'ordinamento e lo rimette nella giusta posizione. La funzione deve restituire il puntatore al primo elemento della lista così ottenuta. Per semplicità, si assuma che tutti gli elementi della lista abbiano valori distinti.

[Esercizio 3 - punti 8]
RoboCook è un frullatore programmabile con forno integrato in grado di cucinare da solo utilizzando fino ad un massimo di 32 ingredienti. Ad ogni minuto RoboCook può aggiungere un nuovo ingrediente oppure continuare a cuocere gli ingredienti che sono stati aggiunti in precedenza (uno stesso ingrediente non può essere aggiunto più volte; quando viene aggiunto un ingrediente quelli già presenti continuano a cuocere). Per ogni ingrediente sono specificati un tempo minimo e un tempo massimo di cottura: entrambi i tempi sono espressi in minuti e sono numeri interi. Scrivere una procedura `robocook(int n, int imin[32], int imax[32])` che calcoli quante ricette diverse possono essere realizzate con n minuti a disposizione (cioè quante sono le ricette che richiedono al più n minuti). Due ricette sono considerate diverse se differiscono per gli ingredienti, o per l'ordine con cui gli ingredienti sono inseriti, oppure per i tempi di cottura di uno qualsiasi degli ingredienti. I tempi di cottura devono rispettare i tempi minimi e massimi indicati negli array `imin[]` e `imax[]`. Insieme al codice si deve fornire una descrizione sintetica della strategia di risoluzione utilizzata. Suggerimento, Non lasciatevi confondere dal testo: si tratta della solita ricorsione. State attenti però a rispettare i tempi di cottura!

[Esercizio 4 - punti 5+4]
Si consideri il linguaggio $L \subseteq \{a, b\}^*$ formato dalle stringhe della forma

$$a^{n_1} b^{m_1} a^{n_2} b^{m_2} \cdots a^{n_k} b^{m_k}$$

con $k \geq 1$ e tali che

$$\begin{cases} n_i + m_i \equiv 3 \pmod 5 & \text{per } i = 1, 2, \ldots, k \\ m_i + n_{i+1} \equiv 2 \pmod 7 & \text{per } i = 1, 2, \ldots, k-1. \end{cases}$$

Si dimostri che esiste un automa a stati finiti deterministico che riconosce L oppure si dimostri che tale automa non può esistere. Si ripeta poi l'esercizio aggiungendo alle condizioni precedenti il vincolo $m_i \geq m_{i+1}$ per $i = 1, 2, \ldots, k-1$.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compito del 14 Giugno 2007

[Esercizio 1 - punti 5+2]
Si scriva una procedura `int VerificaLanci(int n)`, che, dato un intero pari $n > 0$, simuli una sequenza di lanci casuali Testa/Croce e restituisca il minimo numero di lanci necessario affinché la seguente condizione di terminazione sia verificata: negli ultimi n lanci sono uscite esattamente $n/2$ Testa ed $n/2$ Croce. I punti aggiuntivi saranno assegnati in base all'efficienza della soluzione.

[Esercizio 2 - punti 8]
Si considerino le strutture: `typedef struct el1 {char *stringa; struct el1 *next;} listaS;` e `typedef struct el2 {int val; struct el2 *next;} listaI;` che rappresentano rispettivamente un elemento di una lista di stringhe e di una lista di interi. Scrivere una procedura `listaI *distanze(listaS *alpha)` che data una lista di stringhe `alpha` restituisca una lista di interi `num` così definita: `num` e `alpha` hanno lo stesso numero di elementi; se l'i-esimo elemento di `alpha` non compare nelle posizioni $1, 2, \ldots, i-1$ di `alpha` allora i-esimo elemento di `num` deve essere zero; se l'i-esimo elemento di `alpha` compare nelle posizioni $1 \leq j_1 < j_2 < \cdots < j_k < i$ di `alpha` allora i-esimo elemento di `num` deve essere uguale a $i - j_k$. È ammesso l'uso della funzione `strcmp` per confrontare due stringhe.

[Esercizio 3 - punti 7]
Un labirinto viene rappresentato mediante una matrice contenente solamente valori **0** e **1**: un valore **0** indica un muro mentre un valore **1** indica un passaggio. Si definisca un cammino come un insieme di passaggi adiacenti (cioè con un lato in comune). Si scriva una procedura `int cammino(char **a, int n)` che data una matrice $n \times n$ restituisca 1 se esiste un cammino che collega un passaggio nella colonna 0 ad un passaggio nella colonna $n-1$, e restituisca 0 altrimenti. In altre parole si vuole sapere se esiste un insieme di indici $(x_1, y_1), \ldots, (x_k, y_k)$ tali che: a) per $i = 1, \ldots, k$, $a[x_i][y_i] = \mathbf{1}$, b) per $i = 1, \ldots, k-1$, $|x_i - x_{i+1}| + |y_i - y_{i+1}| = 1$, c) $y_1 = 0 \wedge y_k = n-1$. Insieme al codice si deve fornire una descrizione sintetica della strategia di risoluzione utilizzata. Suggerimento: spendete cinque minuti in più per trovare una soluzione semplice piuttosto che buttarvi subito a scrivere la prima soluzione (complicata) che vi viene in testa.

[Esercizio 4 - punti 6+4]
Miriam possiede una scorta illimitata di mattoncini Lego™ rossi, gialli, e blu, ma preferisce quelli rossi. Scrivere una procedura `int rgb(int n)` che dato un intero n, restituisca il numero di torri (sequenze di mattoncini) di altezza n, tali che per $k = 1, 2, \ldots, n$ si abbia che tra i primi k mattoncini il numero di quelli rossi sia almeno la metà del totale. Ad esempio per $n = 3$ la procedura deve restituire 5 in quanto le possibili sequenze sono RRR, RRB, RRG, RBR, RGR. I punti addizionali saranno assegnati a chi fornirà sia la soluzione ricorsiva (che non utilizza array o matrici) che la soluzione basata sulla programmazione dinamica (che esegue un numero di operazioni al più proporzionale a n^2). Insieme al codice si deve fornire una descrizione sintetica della strategia di risoluzione utilizzata.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compitino del 29 Maggio 2007

[Esercizio 1 - punti 7]
Sia dato un array `int *a` di lunghezza $2^h - 1$, con $h > 0$. L'array `a` viene usato per rappresentare un albero binario di ricerca di profondità massima $h-1$: il valore associato alla radice si trova in $a[0]$, e i valori associati ai figli del nodo $a[i]$ si trovano nelle posizioni $a[2i+1]$ e $a[2i+2]$. I valori associati ai nodi sono strettamente positivi; un valore $a[j] = 0$ indica che il nodo j e i suoi discendenti non sono presenti nell'albero. Si scriva una funzione `int ricerca(int *a, int h, int val)` che, dato l'array `a` di $2^h - 1$ elementi con $h > 0$, cerca l'elemento dell'albero di valore pari a `val`, e restituisca l'indice dell'array contente tale valore in caso di esito positivo della ricerca, e restituisca -1 in caso di esito negativo. Per semplicità, si supponga che tutti gli elementi di `a` siano distinti.

[Esercizio 2 - punti 9]
Sia L una lista di stringhe ordinate per lunghezze decrescenti (ogni elemento della lista ha una lunghezza minore o uguale della lunghezza del suo predecessore). Si scriva una procedura `lista *trasposta(lista *)` che, dato un puntatore ad L, costruisca una nuova lista di stringhe tale che l'i-esima stringa della lista è formata concatenando l'i-esimo carattere di ogni stringa di L di lunghezza maggiore o uguale a i. La procedura deve restituire il puntatore al primo elemento della nuova lista così ottenuta. Ad esempio, se L = {`cammello, albero, sasso, ora`}, la lista restituita da `trasposta` deve essere {`caso, alar, mbsa, mes, ero, lo, l, o`}.

[Esercizio 3 - punti 6]
Si consideri la struttura `typedef struct {double inizio; double fine;} intervallo;` che rappresenta un intervallo della retta reale. Si scriva una procedura `unione(intervallo *a, int n)` che dato un array `a[]` contenente n intervalli $a_0, a_1, \ldots a_{n-1}$, calcoli la lunghezza totale degli intervalli che appartengono $\cup_{i=0}^{n-1} a_i$. Suggerimento: considerare la partizione della retta reale indotta dai $2n$ estremi degli intervalli $a_0, a_1, \ldots a_{n-1}$.

[Esercizio 4 - punti 10]
Alice deve partire per una vacanza e vuole caricare sui suoi due lettori `mp3` parte della propria collezione di brani musicali. I lettori possono contenere rispettivamente 200 e 300 minuti di musica e la collezione di Alice consiste in n brani di durata rispettivamente $d_0, d_1, \ldots, d_{n-1}$ minuti. Ad ogni brano Alice ha assegnato un indice di gradimento g_i e vuole caricare sui lettori i brani che massimizzano il suo gradimento totale. Alice però ha dei gusti musicali molto sofisticati e di conseguenza ha definito anche un indice di gradimento congiunto h_{ij} che rappresenta il gradimento di avere i brani i e j sullo stesso lettore. Scrivere una procedura `void Alice(int *d, int *g, int **h, int n)` che dati i vettori delle durate e del gradimento e la matrice (simmetrica) del gradimento congiunto stampi la selezione di brani che massimizza il gradimento totale. In altre parole, si devono trovare due sottoinsiemi I_1, I_2 di $\{0, 1, \ldots, n-1\}$ che massimizzino la quantità

$$\sum_{i \in I_1 \cup I_2} g_i \; + \sum_{i,j \in I_1, i<j} h_{ij} \; + \sum_{i,j \in I_2, i<j} h_{ij}$$

rispettando i vincoli $\sum_{i \in I_1} d_i \leq 200$, e $\sum_{i \in I_2} d_i \leq 300$. Suggerimento: Per ogni brano le possibilità sono: escluderlo, caricarlo nel primo lettore, caricarlo nel secondo, o caricarlo in entrambi.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compitino del 19 Aprile 2007 (file pari)

[Esercizio 1 - punti 7]
Definiamo un gioco di carte, chiamato CartaPiuAlta, come segue. Abbiamo due giocatori, P1 e P2, ed un mazzo di 40 carte (10 carte per ognuno dei quattro semi). In un turno di gioco, P1 e P2 estraggono una carta dal mazzo (senza reimbussolamento). Il giocatore che ha estratto la carta di valore più alto si aggiudica il punto, se le carte hanno lo stesso valore il punto non viene assegnato (i valori delle carte vanno da 1 a 10). Il gioco continua fino a che il mazzo non termina (20 turni di gioco). Al termine del gioco, vince il giocatore con punteggio più alto, oppure con un pareggio nel caso entrambi i giocatori abbiano totalizzato lo stesso punteggio. Scrivere una procedura `int CartaPiuAlta()` che simula una mano del gioco di CartaPiuAlta, e restituisce 0 in caso di pareggio, 1 in caso di vittoria di P1, e 2 in caso di vittoria di P2.

[Esercizio 2 - punti 4+2]
Sia data una matrice quadrata A di ordine $n < 100$ ad elementi 0/1, dove n è un intero pari. La matrice A è detta *a croce di ordine* 2 se e solo se $A[i][j] = 1 \Rightarrow (|i-j| < 2) \vee (|n-1-i-j| < 2)$. Si scriva una procedura `int TrovaOrdine (int n, int A[100][100])` che, data una matrice A definita come sopra, restituisce 1 se A è a croce di ordine 2, e 0 altrimenti. I punti addizionali verrano assegnati in base all'efficienza della soluzione proposta.

[Esercizio 3 - punti 6+3]
Scrivere una procedura `int piccoli(int a[], int n)` che, dato un array `a` di $n > 0$ elementi, restituisce il numero di volte che compare il penultimo elemento (si intende penultimo in ordine di grandezza). Ad esempio se l'array `a[]` contiene gli elementi `{5 1 9 6 3 3 1 3 1 9 3 9 7}` la procedura deve restituire il valore 4 in quanto il penultimo elemento (il 3) compare 4 volte. I punti addizionali verrano assegnati per una soluzione che legge ogni elemento di `a[]` una sola volta.

[Esercizio 4 - punti 5+5]

1. Si costruisca un automa a stati finiti che riconosca il linguaggio $L \subseteq \{a,b,c\}^*$ contenente le stringhe tali che $2^{\#a} + \#b - \#c \equiv 3 \bmod 5$, dove $\#x$ indica il numero di occorrenze del carattere x nella stringa. Esempi di stringhe appartenenti al linguaggio sono *babbab* (in quanto $2^2 + 4 \equiv 3 \bmod 5$, e *cbab* (in quanto $2^1 + 2 - 1 \equiv 3 \bmod 5$).
2. Si consideri ora il linguaggio $L' \subseteq \{a,b,c\}^*$ contenente le stringhe tali che $2^{\#a} + \#b - \#c \equiv 3 \bmod 5$ (come prima) e tali che $\#b + \#c \geq 2^{\#a}$. Si costruisca un automa a stati finiti che riconosca L' o si dimostri che tale automa non può esistere.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compito del 18 Settembre 2006

[Esercizio 1 - punti 5]
Scrivere una procedura `void somma(int *a, int n, int s)` che, dato un array `a` di n numeri interi non negativi $a_0, \ldots, a_{n-1}$ e un intero s, restituisca l'elenco di tutti i sottointervalli $[i, j]$ tale che $a_i + a_{i+1} + \cdots + a_j = s$. La valutazione terrà conto della semplicità ed efficienza della soluzione proposta.

[Esercizio 2 - punti 7]
Sia data una struttura `typedef struct lst {int valore; struct lst *succ;} elemento;` che rappresenta un elemento di una lista di interi. Il campo `valore` rappresenta il valore dell'elemento, ed è compreso fra -100 e 100 (estremi inclusi); il campo `succ` è il puntatore all'elemento successivo della lista. Si scriva la funzione `elemento *RimuoviDoppioni (elemento *p)` che, dato il puntatore ad una lista *non ordinata* di interi, restituisca il puntatore alla lista ottenuta da `p` rimuovendo (e deallocando) tutti i doppioni (elementi il cui campo `valore` è già presente nella lista). Suggerimento: si consiglia di risolvere il problema facendo una sola passata sui dati ed eventualmente utilizzando uno o più array ausiliari (ma meno memoria utilizzate meglio è).

[Esercizio 3 - punti 5+5]
(1) In una villa ci sono k oggetti preziosi di valore rispettivamente $v_0, v_1, \ldots, v_{k-1}$ e di peso rispettivamente $p_0, p_1, \ldots, p_{k-1}$ (valori e pesi sono interi positivi). Un ladro ha bisogno di procurarsi oggetti per un valore complessivo di almeno V unità, ma essendo pigro non vuole rubare più di 5 oggetti e tra tutti i possibili gruppi di ≤ 5 oggetti di valore $\geq V$ vuole selezionare il gruppo con il minor peso complessivo. Scrivere una procedura `void ladropigro(int *v, int *p, int k, int V)` calcoli quale deve essere la scelta di oggetti del ladro, o stampi un messaggio che avverta che tale scelta non è possibile.
(2) Come il punto precedente, ma eliminando il vincolo che gli oggetti non devono essere più di 5. In altre parole si deve determinare, se esiste, tra tutte le combinazioni di oggetti valore $\geq V$ quella di peso minimo.

[Esercizio 4 - punti 5]
Si determini una grammatica di tipo uno che generi il linguaggio $L \subseteq \{a\}^*$ composto dalle stringhe della forma a^{2^n+n} con $n > 0$.

[Esercizio 5 - punti 5]
Si determini un automa a stati finiti che riconosca il linguaggio $L \subseteq \{a, b\}^*$

$$L = \{\alpha | \alpha \text{ non contiene la stringa } aba \quad \textbf{e} \quad 3a_\alpha - 8b_\alpha \equiv 2 \pmod 5\}$$

dove a_α (risp. b_α) indica il numero di a (risp. b) in α.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compito del 17 Luglio 2006

[Esercizio 1 - punti 4+2]
Si scriva una funzione `int **SommePrefisse(int **a, int n)` che, data una matrice quadrata `a` di dimensione `n`> 0, restituisca una matrice quadrata `b` di ordine `n` i cui elementi sono così calcolati:

$$b_{ij} = \begin{cases} a_{ij} & \text{se } i = j \\ \sum_{k=i}^{j} a_{ik} & \text{se } i < j \\ \sum_{k=j}^{i} a_{kj} & \text{se } i > j \end{cases} .$$

I punti aggiuntivi saranno assegnati per l'efficienza della soluzione proposta.

[Esercizio 2 - punti 7]
Siano date le strutture `typedef struct nodo {int val; struct elemento *figli} albero;` e `typedef struct elemento {albero *figlio; struct elemento *proxFiglio} listaFigli` utilizzate per rappresentare un albero in cui ogni nodo può avere un numero arbitrario di figli. Ogni nodo N dell'albero è rappresentato da una struttura `albero`, dove `val` contiene un valore intero associato al nodo, e `figli` contiene un puntatore alla lista dei figli di N (tale puntatore vale `NULL` se N è una foglia). Ogni elemento della lista dei figli contiene il campo `figlio`, che punta ad uno dei figli di N, e dal campo `proxFiglio`, che punta all'elemento successivo nella lista dei figli. Scrivere una procedura `int SommaFoglie(albero *radice)` che dato il puntatore alla radice dell'albero restituisce la somma dei valori contenuti nelle foglie dell'albero. [Suggerimento: se N non è una foglia, le foglie che discendono da N sono date dall'unione delle foglie che discendono dai figli di N].

[Esercizio 3 - punti 6]
Scrivere una funzione `int sottosequenza(char *s, char *t)` che date due stringhe (array di caratteri terminati dal carattere `\0`) $s = s_0 \cdots s_{n-1}$, e $t = t_0 \cdots t_{m-1}$ restituisca 1 se i caratteri di s formano una sottosequenza di t e restituisca 0 altrimenti. In altre parole, la procedura deve restituire 1 se e solo se esistono $0 \leq \ell_0 < \ell_1 < \cdots < \ell_{n-1} < m$ tali che per $i = 0, \ldots, n-1$ si ha $s_i = t_{\ell_i}$.

[Esercizio 4 - punti 6]
Si costruisca un automa che riconosca il linguaggio $L \subseteq \{\mathbf{0}, \mathbf{1}, \mathbf{2}\}^*$ composto dalle stringhe non vuote che a) terminano per **1** o **2**, b) interpretate come numeri in base tre, *letti a partire dalla cifra meno significativa*, risultano essere congrui a 3 modulo 4. Esempi di stringhe appartenenti al linguaggio sono: $\mathbf{01} = 0 \cdot 3^0 + 1 \cdot 3^1 = 3$, e $\mathbf{201} = 2 \cdot 3^0 + 0 \cdot 3^1 + 1 \cdot 3^2 = 11$.

[Esercizio 5 - punti 7]
Si determini una grammatica di tipo 2 che generi il linguaggio sull'alfabeto $\{a, b, +, =\}^*$ composto dalle stringhe della forma $x^i + y^j = z^k$ con $x, y, z \in \{a, b\}$ e $i, j, k \geq 0$ che soddisfano alle seguenti condizioni: 1) se $x = y$ allora $z = x$ e $k = i + j$, 2) se $x \neq y$ e $i \geq j$ allora $z = x$ e $k = i - j$, 3) se $x \neq y$ e $i < j$ allora $z = y$ e $k = j - i$. Esempi di stringhe appartenenti al linguaggio sono: $b + bb = bbb$, $aaa + aaaaa = aaaaaaaa$, $a + b =$, $aa + bbbbb = bbb$, e $aaaa + b = aaa$.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compito del 20 Giugno 2006

[Esercizio 1 - punti 6]
I numeri di Fibonacci di ordine n, con $n > 0$, sono così definiti:

$$f_n(0) = a_0,\ f_n(1) = a_1,\ \ldots,\ f_n(n-1) = a_{n-1}, \qquad f_n(i) = \sum_{j=1}^{n} f_n(i-j) \quad \text{for } i \geq n,$$

dove $a_0, \ldots, a_{n-1}$ sono detti termini iniziali. Si scriva una funzione `int nFibonacci(int a[],int n, int i)` che, dato l'ordine `n` della successione di Fibonacci ed il vettore dei termini iniziali `a[]`, calcola l'`i`-esimo termine della successione di Fibonacci di ordine `n`. La valutazione terrà conto della semplicità ed efficienza della soluzione proposta.

[Esercizio 2 - punti 8]
Sia data la struttura `typedef struct elemento {char *cognome; int mat; struct elemento *proxCogn; struct elemento *proxMat} elenco;` che rappresenta un elemento di una lista di studenti. Gli studenti sono ordinati secondo due criteri: *i*) per cognome, in ordine alfabetico, tramite il puntatore `proxCogn`, e *ii*) per numero di matricola, in senso crescente, tramite il puntatore `proxMat`. Si scriva una funzione `elenco **insStudente(elenco *a[2], char *s, int matr)` che, dato *a*) un vettore di due puntatori, che puntano rispettivamente al primo studente in ordine alfabetico (`a[0]`), ed allo studente con numero di matricola più piccolo (`a[1]`); *b*) una stringa `s` che contiene il cognome dello studente da inserire, e *c*) un intero `matr` che rappresenta il numero di matricola dello studente da inserire, crea ed inserisce nella lista un nuovo elemento corrispondente allo studente, mantenendo l'ordinamento per cognome (tramite i puntatori `proxCogn`) e per numero di matricola (tramite i puntatori `proxMat`). La funzione deve restituire un array di puntatori, contente in posizione `0` il puntatore al primo studente in ordine alfabetico, ed in posizione `1` il puntatore allo studente con numero di matricola più piccolo.

[Esercizio 3 - punti 6]
Scrivere una procedura `void totocalcio(int n, int max1, int max2, int max3)` che stampi tutte le stringhe distinte di lunghezza `n` formate dai caratteri $\{\mathbf{1}, \mathbf{2}, \mathbf{X}\}$, che contengono non più di `max1` **1**, non più di `max2` **2**, e non più di `max3` **X** (tutti e tre i vincoli devono essere soddisfatti).

[Esercizio 4 - punti 5+3]

1. Si determini una grammatica di tipo 2 che generi il linguaggio su $\{a, b, c\}^*$ composto dalle stringhe della forma $a^i b^j c^k$ con i, j e k interi positivi, $i + k \equiv 1 \pmod 3$ e $j \leq k$.
2. Si dimostri che il linguaggio definito al punto 1 non può essere generato da una grammatica di tipo 3.

[Esercizio 5 - punti 5]
Si consideri il linguaggio $L \subseteq \{a, b, c\}^*$ composto dalle stringhe della forma $a^i b^j c^k$ dove i, j, k sono interi maggiori di zero tali che $i + 2j \equiv 1 \pmod 5$ e $j + k \equiv 2 \pmod 7$ (entrambe le condizioni devono essere soddisfatte). Esempi di stringhe appartenenti al linguaggio sono $a^2b^2c^7$, ab^5c^4, e a^4bc. Si dimostri che esiste un automa a stati finiti deterministico che riconosce L oppure si dimostri che tale automa non può esistere.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compitino del 5 Giugno 2006

[Esercizio 1 - punti 5]
Scrivere una procedura `double *convc(double *x, double *y, int n)` che dati due array x, y entrambi di lunghezza n, allochi e restituisca l'array z, sempre di lunghezza n, contenente la convoluzione ciclica di x e y. Gli elementi di z sono definiti dalla relazione:

$$z_i = \sum_{j=0}^{n-1} x_j y_{(i-j) \bmod n}, \qquad \text{per } i = 0, 1, \ldots, n-1$$

dove $t \bmod n$ indica il resto della divisione fra t e n.

[Esercizio 2 - punti 6]
Sia data una struttura `typedef struct nodo {struct nodo *sx; struct nodo *dx;} albero;` che rappresenta un albero binario. Si scriva una procedura `struct nodo *insguidato(albero *r, int scelte[], int n)` che dato un puntatore alla radice di un albero binario e un vettore `scelte[]` di `n` interi inserisca un nuovo nodo secondo la seguente procedura. Il nuovo nodo deve essere inserito come figlio di una foglia dell'albero. Per individuare tale foglia si utilizza il vettore `scelte[]`. Partendo dalla radice, in presenza di una situazione di scelta nello scorrere gli elementi dell'albero (cioè, quando l'elemento considerato ha più di un figlio), si deve seguire il puntatore destro `dx` se il numero contenuto nella posizione corrente del vettore `scelte` è 0, il puntatore sinistro `sx` altrimenti. Simile criterio, controllato dal vettore `scelte`, deve essere seguito per collegare il nuovo nodo come figlio destro o sinistro della foglia selezionata. Il vettore delle scelte deve essere scandito a partire dal primo elemento, e letto ciclicamente nel caso in cui il numero di scelte da effettuare sia maggiore di `n`. La funzione deve restituire il puntatore alla radice del nuovo albero così formato.

[Esercizio 3 - punti 7]
Sia data la struttura `typedef struct elem {char *s; struct elem *next;} elenco;` che rappresenta un elemento di una lista di stringhe, dove `next` è il puntatore all'elemento successivo della lista. Si scriva una procedura `elenco *spezza(char *s)` che data una stringa di caratteri `s[]` restituisca la lista ordinata delle parole che appaiono in `s[]` (l'ordine delle parole nella lista deve quindi essere lo stesso con cui appaiono nella stringa). Una parola è definita come una sequenza non vuota di caratteri diversi dallo spazio. Non è ammesso l'uso di procedure di libreria per le operazioni sulle stringhe.

[Esercizio 4 - punti 6+3]
Una mappa viene rappresentata mediante una matrice $n \times n$ di interi positivi $a[\cdot][\cdot]$ tale che $a[i][j]$ indica il costo di entrare nella casella (i, j). Scrivere una procedura `int raggiungibile(int **a, int n, int x, int y, int maxcosto)` che data una mappa $a[n][n]$ e due interi x, y con $0 \leq x, y < n$, restituisca 1 se esiste un percorso di costo minore di `maxcosto` dalla casella $(0, 0)$ alla casella (x, y), e restituisca 0 se tale percorso non esiste. I punti extra sono assegnati per soluzioni particolarmente efficienti. [Suggerimento: esiste una soluzione semplice basata sulla ricorsione].

[Esercizio 5 - punti 3+3]

1. Si determini una grammatica di tipo uno che generi il linguaggio su $\{a, b, c, d\}^*$ composto dalle stringhe della forma $a^n b^m c^n d^m$ con n e m interi positivi.
2. Si determini una grammatica di tipo uno che generi il linguaggio su $\{a, b, c, d, e\}^*$ composto dalle stringhe della forma $a^n b^m c^n d^m e^n$ con n e m interi positivi.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compitino del 27 Aprile 2006

[Esercizio 1 - punti 7]
Scrivere una procedura `int simula_dado(int a, int b)` che simula una successione pseudo-casuale di lanci di dado terminando quando la somma degli ultimi sei lanci di dado è compresa fra a e b (estremi inclusi). La procedura deve stampare l'esito ed il numero progressivo di ogni lancio del dado, e restituire il numero di lanci che sono stati necessari per verificare la condizione di terminazione.

[Esercizio 2 - punti 7]
Due interi positivi si dicono *amicabili* se la somma dei divisori del primo è uguale al secondo e viceversa (ad esempio 220 e 284 sono amicabili). Scrivere una procedura `int amicabili(int a, int b)` che restituisca 1 se a e b sono amicabili, e restituisca 0 altrimenti.

[Esercizio 3 - punti 7]

Scrivere una procedura `int max_diff(int a[10][10], int` n`)` che prenda in input una matrice di interi $n \times n$ con $n \leq 10$ e restituisca il massimo del valore assoluto delle differenze di due elementi nella stessa riga o nella stessa colonna. Ad esempio, se la matrice di ingresso è

5	4	8	2	3
8	5	3	1	10
4	7	3	1	2
6	12	3	1	10
2	6	3	16	3

la massima differenza è 15 (ottenuta nella quarta colonna).

[Esercizio 4 - punti 4]
Si determini un automa a stati finiti che riconosca il linguaggio $L \subseteq \{a, b\}^*$

$$L = \{\alpha | \alpha \text{ non contiene le sottostringhe } bbb \text{ o } aa \quad \textbf{e} \quad a_\alpha + 2b_\alpha \equiv 1 \pmod 3\}$$

dove a_α (risp. b_α) indica il numero di a (risp. b) in α.

[Esercizio 5 - punti 7]
Si dimostri che il linguaggio $L \subseteq \{0, 1, \ldots, 9\}^*$ contenente la rappresentazione decimale di $n!$ per $n \in \mathbb{N}$ (quindi $L = \{1, 2, 6, 24, 120, \ldots\}$) non è un linguaggio regolare.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compito del 13 Febbraio 2006

[Esercizio 1 - punti 7]
Sia data una matrice `a` di 10 colonne ed `n` righe i cui elementi possono essere soltanto `0` o `1`. Una sottomatrice 2×2 di `a` viene detta di tipo i se e solo se contiene i elementi uguali ad 1. Si scriva una funzione `int *contaSottomatrici(int a[][10], int n)` che, data una matrice `a` definita come sopra, restituisce un array `contaTipi` di 5 elementi interi, dove `contaTipi[i]` contiene il numero di sottomatrici di tipo i presenti nella matrice `a`.

[Esercizio 2 - punti 8]
Scrivere una funzione `int Poker();` che simula una mano semplificata del gioco del poker. La funzione deve estrarre casualmente cinque carte da un mazzo di 52 carte, e deve restituire il valore della massima combinazione ottenuta con le cinque carte estratte. Le possibili combinazioni e i rispettivi valori sono i seguenti.

- Poker (quattro carte con lo stesso valore) **5**;
- Colore (cinque carte dello stesso seme) **4**;
- Tris (tre carte con lo stesso valore) **3**;
- Doppia coppia (due carte con lo stesso valore) **2**;
- Coppia (due carte con lo stesso valore) **1**;
- nessuna combinazione **0**.

Per semplicità, si assuma che il seme di una carta sia un valore intero compreso fra 0 e 3, ed il valore di una carta sia un intero compreso fra 1 e 13.

[Esercizio 3 - punti 4+4]
Scrivere una funzione `void quaterne(int *a, int n, int s)` che, dato un array `a` di n numeri interi non negativi $a_0, \ldots, a_{n-1}$, e un intero s stampi l'elenco di tutte le quaterne di indici $\langle i_1, i_2, i_3, i_4 \rangle$ tali che

$$0 \leq i_1 < i_2 < i_3 < i_4 < n, \quad \text{e} \quad a_{i_1} + a_{i_2} + a_{i_3} + a_{i_4} = s.$$

Si scriva poi una funzione `int pari_uple(int *a, int n, int s)` che restituisca 1 se esiste un insieme non vuoto di indici $\langle i_1, i_2, \ldots, i_{2m} \rangle$ tali che

$$\sum_{k=1}^{2m} a_{i_k} = s$$

e restituisca 0 altrimenti. [In altre parole, invece di cercare quaterne di numeri con somma s cerchiamo $2m$-uple di numeri con somma s.]

[Esercizio 4 - punti 3+6]

1. Si determini una grammatica di tipo due che generi il linguaggio su $\{a, b, c\}^*$ composto dalle stringhe della forma $a^i b^j c^k$ con i, j, k positivi e $|i - k| \leq 3$.
2. Si determini una grammatica di tipo due che generi il linguaggio su $\{a, b, c\}^*$ composto dalle stringhe della forma $a^i b^j c^k$ con i, j, k positivi e $|i - k| > j$. Si dimostri poi che tale linguaggio non può essere generato da una grammatica di tipo 3.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compito del 13 Gennaio 2006

[Esercizio 1 - punti 6]
Sia data la struttura `typedef struct nodo {int v; struct nodo *left; struct nodo *right;} albero;` che rappresenta un *albero binario* di valori interi. La profondità di un albero è definita come la distanza massima di una foglia dell'albero dalla radice (un albero contenente solo la radice ha quindi profondità zero). Si scriva una funzione `int CalcolaProfondità(albero *a)` che, dato il puntatore alla radice di un albero, calcola la profondità dell'albero e ne restituisce il valore (si può assumere che sia `a!=NULL`).

[Esercizio 2 - punti 6]
Sia data la struttura `typedef struct persona {char *nome; char *cognome; struct persona *succ;} elenco;` che rappresenta un elemento di una lista di nominativi, dove `succ` è il puntatore all'elemento successivo della lista. Si scriva una funzione `int trova(elenco *lista, char *sn, char *sc)` che dato il puntatore a una lista di nominativi e le due stringhe `sn` e `sc` restituisca il numero di persone il cui nome termina per `sn` e il cognome termina per `sc`. È ammesso l'uso delle funzioni di libreria `int strcmp(char *s, char *t)` (confronto lessicografico fra stringhe) e `int strlen(char *s)` (calcolo della lunghezza di una stringa).

[Esercizio 3 - punti 5+6]
Scrivere una procedura `int num_cime(int *a, int n)` che dato un array `a[]` contenente gli interi $a_0, \ldots, a_{n-1}$, restituisca il numero di cime di `a`. Una cima è costituita da una coppia di indici (i, j) tali che

$$i < j, \qquad a_i = a_{i+1} = \cdots = a_j, \qquad i > 0 \Rightarrow a_i > a_{i-1}, \qquad j < n-1 \Rightarrow a_j > a_{j+1}$$

(in altre parole si tratta di un massimo locale assunto in almeno due valori consecutivi). Si scriva poi una procedura `int num_quote(int *a, int n)` che restituisca il numero di quote distinte raggiunte dalle cime di `a`. Ad esempio, se `a` = [5 1 9 9 9 3 4 4 2 9 9 5] ci sono tre cime ([9 9 9], [4 4], [9 9]) ma il numero di quote distinte è 2 (9 e 4). Per la realizzazione della seconda parte è ammesso l'utilizzo di un array ausiliario. La valutazione terrà conto della semplicità ed efficienza delle soluzioni proposte.

[Esercizio 4 - punti 4+5]

1. Si costruisca un automa a stati finiti che riconosca il linguaggio composto dalle stringhe binarie che a) iniziano per 1, b) interpretate come numero binario (nella maniera usuale) hanno un valore $\equiv 2 \bmod 5$, c) non contengono due 1 consecutivi. Ad esempio, la stringa 100101 appartiene al linguaggio in quanto corrisponde al valore decimale $37 = 2^5 + 2^2 + 2^0$.
2. Si dimostri che non esiste un automa a stati finiti che riconosce il linguaggio composto dalle stringhe binarie che a) iniziano per 1, b) interpretate come numero binario (nella maniera usuale) hanno un valore $\equiv 2 \bmod 5$, c) il numero massimo di 1 consecutivi è minore o uguale al numero massimo di zeri consecutivi. Un esempio di stringa appartenente a questo linguaggio è 10010011 che corrisponde al valore decimale 147.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compito del 9 Settembre 2005

[Esercizio 1 - punti 6]
Il gioco del "7 e mezzo" si gioca con 40 carte; le carte da 1 a 7 hanno il corrispondente valore, mentre quelle da 8 a 10 valgono mezzo punto. Il punteggio di un insieme di carte è dato dalla somme dei valori delle singole carte, ma se questa somma supera 7.5 il punteggio è 0. Scopo del gioco è avvicinarsi il più possibile al punteggio di 7.5. Si scriva una funzione `float SetteEMezzo(void)` che simuli una mano del gioco. La procedura deve: a) estrarre una carta dal mazzo, b) stampare il punteggio attuale e chiedere al giocatore se vuole un'altra carta fino a quando il giocatore non decide di fermarsi oppure la somma dei valori supera 7.5. La funzione deve restituire il punteggio realizzato dal giocatore.

[Esercizio 2 - punti 7]
Sia data una struttura `typedef struct nodo {int valore; struct nodo *left; struct nodo *right;} albero;` che rappresenta un nodo di un *albero binario ordinato* di valori interi. Dato un elemento dell'albero di valore v, tutti gli elementi nel sottalbero sinistro hanno valore minore di v, mentre tutti gli elementi del sottoalbero destro hanno valore maggiore di v (si assume che tutti i valori siano distinti). Si scrivano le seguenti funzioni:

- ◇ `void StampaOrdinata (albero *a)` che, dato il puntatore alla radice di un albero ordinato, stampa il valore degli elementi contenuti nell'albero, in ordine crescente.
- ◇ `albero *CercaElemento (albero *a, int num)` che, dato il puntatore alla radice di un albero ordinato, ed un valore intero `num`, verifica se nell'albero esiste un elemento con valore uguale a `num`, e restituisce il puntatore a tale elemento. Nel caso in cui nessun elemento dell'albero abbia valore uguale a `num`, la funzione deve restituire il puntatore nullo `NULL`.

[Esercizio 3 - punti 7]
Scrivere una funzione `int somma9(int *a, int n)` che, dato un array `a` di n numeri interi non negativi $a_0, \ldots, a_{n-1}$, restituisca 1 se esistono k elementi consecutivi $a_i, \ldots, a_{i+k-1}$ la cui somma sia 9. La valutazione terrà conto della semplicità ed efficienza della soluzione proposta.

[Esercizio 4 - punti 7]
Si costruisca un automa che riconosca il linguaggio $L \subseteq \{0,1,2\}^*$ composto dalle stringhe non vuote che interpretate come numeri in base tre risultano essere multipli o di 5, o di 7 o di 9.

[Esercizio 5 - punti 5]
Si dimostri che il linguaggio $L \subseteq \{0,1\}^*$ composto da tutte e sole le stringhe binarie il cui valore ha la forma $(1+2^k)^2$ per $k = 0, 1, \ldots$ non è regolare.

Università di Pisa
Corso di Laurea in Matematica

Fondamenti/Laboratorio di Programmazione Compito del 12 Luglio 2005

[**Esercizio 1 - punti 4**]
Scrivere una funzione `int *KSomme(int *a, int n, int k)` che, dato un array `a` di n numeri interi $a_0, \ldots, a_{n-1}$, ed un parametro intero $0 < k < n$, restituisca il puntatore ad un array `somme` di n interi, dove

$$\texttt{somme[i]} = \sum_{j=0}^{k-1} a_{i+j \bmod n}.$$

Ad esempio, se invocata sull'array `a=`{`1,5,3,8,9,15`} con parametro $k = 3$ la funzione `KSomme` deve restituire un puntatore all'array: {`9,16,20,32,25,21`}. La procedura deve restituire `NULL` se l'input non è valido, o non è disponibile la memoria per la creazione dell'array `somme`.

[**Esercizio 2 - punti 5+6**
Sia data una struttura `typedef struct persona {char *nome; char *cognome; struct persona *succ; struct persona *prec;} elenco;` che rappresenta un elemento di una lista di nominativi, dove `succ` è il puntatore all'elemento successivo della lista, e `prev` è il puntatore all'elemento precedente della lista. Si scrivano le seguenti funzioni:

1. `elenco *InsNominativo(elenco *lista, char *Nome, char *Cognome)` che, dati il puntatore ad una lista ordinata di elementi di tipo `elenco` e due stringhe corrispondenti al nome ed al cognome del nominativo da inserire, crea un nuovo elemento di tipo `elenco` e lo inserisce nella lista mantenendo l'ordinamento. La funzione restituisce un puntatore alla nuova lista così creata. Il criterio di ordinamento è lessicografico, ordinando per cognome e, a parità di cognome, per nome.
2. `int ContaOmonimi(elenco *lista)` che, dato il puntatore ad una lista ordinata di elementi di tipo `elenco`, restituisce il numero di omonimi (cioè persone con lo stesso nome e cognome) presenti nella lista.

È ammesso solo l'uso della funzione di libreria `int strcmp(char *s, char *t)` per il confronto lessicografico fra stringhe.

[**Esercizio 3 - punti 9**]
Scrivere una procedura `int conta(char **a, int n)` che data una matrice di caratteri di dimensione $n \times n$, con $n \geq 4$ contenente soltanto valori **0** e **1** restituisca il numero di sottomatrici 4×4 *distinte* presenti all'interno di a. Si devono considerare solamente le sottomatrici 4×4 formate da righe e colonne contigue (quindi in una matrice $n \times n$ ci sono $(n-3)^2$ sottomatrici 4×4). La valutazione terrà conto della semplicità ed efficienza della soluzione proposta. [Suggerimento: si osservi che il numero massimo di sottomatrici distinte è $2^{16} = 65536$.]

[**Esercizio 4 - punti 4+4**]
Si dimostri che il linguaggio $L \subseteq \{a, b, c\}^*$ formato dalle stringhe della forma $w = c^i(ab)^j a^k b^\ell$ con $|w| \equiv_{13} 11$, $i \geq 4$, e $j + k \equiv_3 1$ è regolare. Si dimostri che il linguaggio non è più regolare se si aggiunge la condizione che nella stringa w il numero di b sia maggiore del numero di a.