Informatica Generale
Divertiamoci giocando con l'Homework 2

docente: Ivano Salvo
Sapienza Università di Roma

pubblicazione: 13.IV.2012 - consegna 3.V.2012

Esercizio 1: (MasterMind) Scrivere un programma che legge in input 20 cifre (una alla volta con `scanf("%d",...);`) e considera le prime dieci come una combinazione segreta del gioco del master mind, e le seconde 10 come un tentativo di indovinare la combinazione segreta. Produrre in output 2 numeri (su due righe diverse), tali che il primo sia il numero delle cifre corrette al posto giusto, e il secondo sia il numero delle cifre corrette, ma al posto sbagliato.

Esempio: Supponiamo venga letta la seguente sequenza di cifre: `1 2 3 4 5 5 4 3 2 1 1 6 7 2 1 5 5 3 1 4`. Va interpretata come segue:

combinazione segreta:	1	2	3	4	5	5	4	3	2	1
tentativo:	1	6	7	2	1	5	5	3	1	4
risultato:	•			○	○	•	○	•		○

dove • significa "numero giusto al posto giusto" e ○ significa "numero giusto al posto sbagliato". Osservate che l'ultimo 1 del tentativo non produce nessun risultato perchè ho già "impegnato" entrambi gli 1 della combinazione segreta. In questo caso il vostro programma dovrebbe rispondere quindi: `3 4`.

Esercizio 2: (Enigmistica: Lucchetto) Nel linguaggio enigmistico un *lucchetto* è un gioco caratterizzato dallo schema $XY + YZ \to XZ$.

Forse un informatico teorico direbbe piuttosto che prese tre parole (o sequenze) w_1, w_2, w_3 su un certo alfabeto di simboli Σ, w_3 è un lucchetto di w_1 e w_2 se e solo se esistono tre parole x, y, z tale che $w_1 = xy$, $w_2 = yz$ e $w_3 = xz$.

Dovete scrivere un programma che legge due sequenze di caratteri w_1 e w_2, ciascuna terminata dal simbolo '`*`' e produce in output il *lucchetto massimale* tra le due sequenze (leggete i caratteri uno alla volta con `scanf("%c",...);`. Non usate stringhe). Il programma deve quindi trovare la *più lunga* sequenza finale di w_1 che è uguale alla sequenza iniziale di w_2, e produrre in output il lucchetto risultante seguendo le regole sopra descritte (senza ovviamente preoccuparsi che il risultato

abbia significato o meno nella lingua italiana). Scrivete l'output scrivendo i caratteri uno alla volta, con l'istruzone `printf("%1c",...);` in modo da non lasciare spazi tra un carattere e l'altro.

ESEMPI: Se le parole di ingresso sono uguali, il risultato è la sequenza vuota. Se non esiste nessun suffisso della prima comune a un prefisso della seconda, allora il risultato è semplicemente la concatenazione delle due parole. Ad esempio se l'input fosse: `r a r a * m e n t e *` il vostro programma dovrebbe scrivere in output `raramente`. Attenzione però che in alcuni casi (palindromia di suffissi e prefissi) il lucchetto non è unico, ma voi dovete restituire il massimale. Ad esempio se l'input fosse `t o n o * o n o r e *` dovete scrivere in output `tre` e non `tonnore`.

⋆Esercizio 3: (SUPERENALOTTO)[1] Dato l'insieme $[n] = \{1, 2, \ldots, n\}$ dei primi n numeri, e fissato k ($0 \leq k \leq n$) è possibile numerare ogni k-upla di numeri in $\mathcal{P}_k([n])$ (cioè l'insieme dei sottoinsiemi di k elementi di $[n]$) con un numero compreso tra 1 e $\binom{n}{k}$ nel seguente modo. Anche se le k-uple sono insiemi, consideriamo ogni k-upla $\{a_1, \ldots, a_k\}$ sempre scritta in forma *canonica* come una sequenza crescente di interi $\langle a_1, \ldots, a_k \rangle$ tali che $a_1 < a_2 < \ldots < a_{k-1} < a_k$. A questo punto, tutte le k-uple possono essere enumerate seguendo l'*ordine lessicografico* (cioè quello del dizionario).

Venendo al SuperEnalotto, in cui vengono considerate sestine formate con i numeri da 1 a 90, la sestina $\langle 1, 2, 3, 4, 5, 6 \rangle$ avrà il numero 1, la sestina $\langle 1, 2, 3, 4, 5, 7 \rangle$ il numero 2, la sestina $\langle 4, 23, 32, 38, 55, 70 \rangle$ il numero 142354920, la sestina $\langle 15, 25, 34, 54, 70, 88 \rangle$ avrà il numero 412620570, fino alla sestina $\langle 85, 86, 87, 88, 89, 90 \rangle$ che avrà il numero 622614630, che correttamente corrisponde a $\binom{90}{6}$.

Dovete scrivere un programma che data una sequenza di numeri interi chiusa da uno 0, interpreta il primo numero come la codifica della sestina vincente, i successivi come la codifica delle sestine giocate e stampa in output il massimo punteggio ottenuto dalle sestine giocate rispetto alla sestina vincente. Suggerisco di:

- scrivere una funzione "efficiente" per calcolarsi i coefficienti binomiali $\binom{n}{k}$ (vedi esercizio nella dispensa "Vettori" per avere un suggerimento);
- scrivere una funzione `void decodifica(int s[], int c, int n, int k)` che restituice nell'array `s` la k-upla di codice c, in $\mathcal{P}_k([n])$;
- scrivere una funzione `int punti(int s1[], int s2[], int k)` che calcola quanti punti realizza la la k-upla $s1$ rispetto a $s2$ (laboratorio, 12.IV.2012);
- non è necessario, ma potrebbe far comodo per testare il programma scrivere la funzione inversa `int codifica(int s[], int n, int k)`.

[1] Questo esercizio ha da ritenersi facoltativo. Chi volesse cimentarsi può trovare ispirazione nel problema (simile) della codifica delle permutazioni (nella dispensa: "Strutture Dati e Codifiche") che sarà trattato nella lezione del 20.IV.2012.