Laboratorio di Fondamenti di Informatica 10/11/2015

Anno accademico 2015/2016

1a. Palindromi

Si definisca una funzione di prototipo

```
int palindromo( char * p );
```

che riceve come parametro una stringa p e restituisce 1 se la stringa è un palindromo, 0 altrimenti.

Una stringa è un palindromo se risulta uguale leggendola da destra a sinistra o da sinstra a destra.

ALA, INNI, KAYAK, AEREA, ESOSE, ANILINA, INGEGNI, AVALLAVA... sono palindromi

SEDANO, TROGOLO, RECRUDESCENZA, RAMARRO, IPALLAGE... invece no!

1b. Palindromi propri e con trasporto

Si legga da stdin una sequenza *a priori illimitata* di parole di almeno due caratteri. Di ogni parola (che si suggerisce di leggere come una stringa, con una sola istruzione `scanf("%s",…)`), si verifichi se è palindroma, e nel caso si stampi un messaggio. La sequenza e l'analisi terminano quando si legge una parola di lunghezza 1. Si verifichi anche se le parole di 3 o più lettere sono palindrome "con trasporto", cioè se si possono leggere anche "al contrario" spostando la prima lettera in fondo oppure spostando l'ultima lettera all'inizio (esempi: **c**ovavo <-> ovavo**c** , ama**i** <--> **i**ama). Al termine, si stampi anche il numero di palindromi propri e con trasporto. Esempi:

```
Tutti sapevano che anna aveva otto radar ma pochi la onorarono !
  "anna" e' palindroma
  "aveva" e' palindroma
  "otto" e' palindroma
  "radar" e' palindroma
  "onorarono" e' palindroma
Hai digitato in totale 6 parole palindrome proprie e 0 con trasporto

La banana assomiglia assai alla patata e poco all' ananas
  "banana" e' palindroma con trasporto
  "assai" e' palindroma con trasporto
  "alla" e' palindroma
  "patata" e' palindroma con trasporto
  "poco" e' palindroma con trasporto
  "ananas" e' palindroma con trasporto
Hai digitato in totale 1 parole palindrome proprie e 5 con trasporto
```

Suggerimento: si noti che una parola p è palindroma con trasporto se e solo se è palindroma la parola ottenuta da p senza considerare la prima oppure l'ultima lettera. Si **riusi**, quindi, la funzione palindromo()!!

2. Cambi di base, reloaded

Si definisce il tipo di dato **`NumeroCodificato`** come segue:

```
typedef struct { int base;
                 char cifre[20] } NumeroCodificato;
```

nel quale la base è un intero compreso tra 2 e 10 e **`cifre`** è una stringa (terminata da '\0') contenente i caratteri corrispondenti alle cifre ('0', '1', …) della codifica del numero in base **`base`**. Si definiscano, codifichino e testino le funzioni:

`int convertiInt(NumeroCodificato n)`

che calcola il valore del numero n e lo restituisce espresso come intero

`void stampaDec()`

che stampa il numero n codificato come numero decimale

`NumeroCodificato codifica(int n, int b)`

che codifica l'intero n in un NumeroCodificato di base b e lo restituisce al chiamante

`NumeroCodificato converti(NumeroCodificato n, int b2)`

che converte il numero n in un altro, codificato in base b2, e lo restituisce al chiamante

3. Allacciamo le parole, basic version (1/2)

- Due parole p_1, p_2 si definiscono *allacciabili* se un suffisso proprio ***s*** di p_1 è anche prefisso proprio di p_2, cioè hanno la forma $p_1 = w_1\mathbf{s}$ $p_2 = \mathbf{s}w_2$ (dove ***s*** è *proprio* se è di almeno un carattere e più corto di p_1 e di p_2).

Esempi di parole allacciabili:

(o**ca**, **ca**rina) ⇒ o**ca**rina

(spo**ra**, **ra**dici) ⇒ spo**ra**dici

(impo**sta**, **sta**zione) ⇒ impo**sta**zione

(corp**o**, **o**razione) ⇒ corp**o**razione

(iso**la**, **la**mento) ⇒ iso**la**mento

(bu**giardi**, **giardi**no) ⇒ bu**giardi**no *(quello dei farmaci)*

(violon**cello**, **cello**phane) ⇒ violon**cello**phane

Esempi di coppie non allacciabili:

(violon**cell**o, **cell**ulare), (**forma**, **forma**zione), (co**raggio**, **raggio**), (trogolo, zangola)

Va da sé che le coppie di parole allacciabili più interessanti sono quelle in cui la parola w_1sw_2 (ottenuta fattorizzando il suf-/pre-fisso) è una parola di senso compiuto, specie se non collegato al significato delle parole di partenza (e.g., "ocarina"). Ma anche l'orrenda coppia (violoncello, cellophane) rispetta la definizione, come del resto anche la coppia (abcde, cdefgh).

3. Allacciamo le parole, basic version (2/2)

- Si codifichino le funzioni

int misuraoverlap(**char** *p1, **char** *p2)

che restituisce la lunghezza del massimo suf-/pre-fisso proprio in comune tra p1 e p2 (cioè la lunghezza di s, se $p1 = w_1 s$ e $p2 = s w_2$)

void stampa_allacciate(**char** *p1, **char** *p2)

che riceve due parole come parametri e le **stampa** a video, allacciate o no, in base alla lunghezza del pre/suf-fisso comune, nel modo seguente:

$(w_1[s)w_2]$ se sono allacciabili,

$(p_1)[p_2]$ se non lo sono.

Esempi:

stampa_allacciate("care", "restia")	→	**(ca[re)stia]**
stampa_allacciate("poste", "osteggiare")	→	**(p[oste)ggiare]**
stampa_allacciate("trogolo", "zangola")	→	**(trogolo)[zangola]**
stampa_allacciate("coraggio", "raggio")	→	**(coraggio)[raggio]**
stampa_allacciate("AB", "BC")	→	**(A[B)C]**

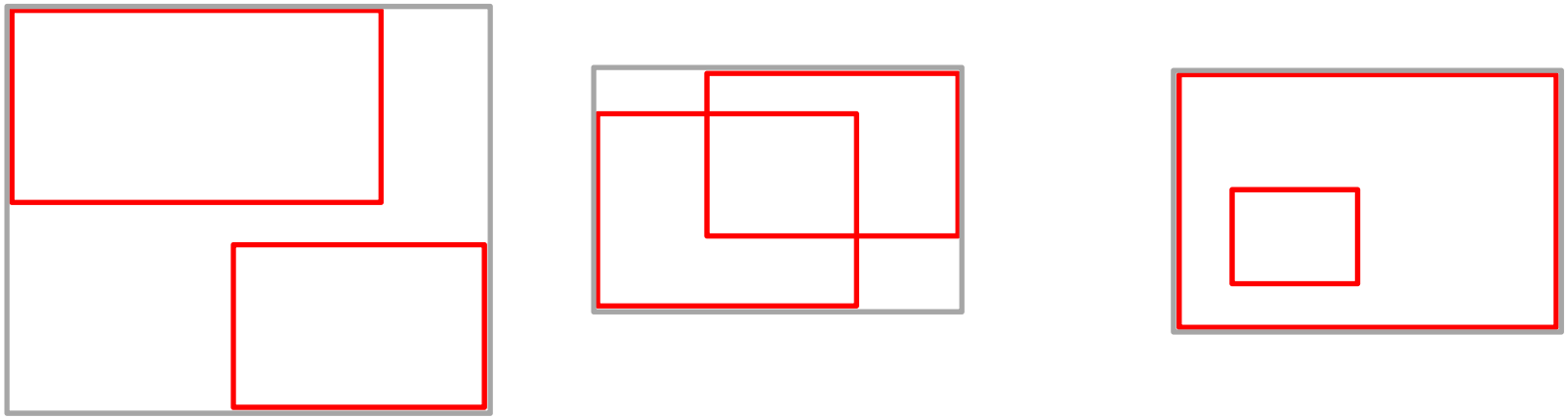
4. Rettangoli "dritti", reloaded

- Chiamiamo Rect i rettangoli coi lati paralleli agli assi, rappresentati come struct contenenti solo due punti (rispettivamente i vertici di *nord-ovest* e di *sud-est*).
- Si codifichino le funzioni
  - **leggiRect()** che restituisce il Rect definito da due punti inseriti dall'utente (controllando che sia non-degenere, con il secondo punto inserito a sud-est del primo punto)
  - **stampaRect(…)** che "stampi" a video il rettangolo nella forma **[2.50 ; 8.00]**
[4.34 ; 2.45]
  - **Rect inviluppo(Rect r1, Rect r2)** che, dati due Rect r1 e r2, restituisce un Rect che ne è l'inviluppo (definizione di inviluppo richiamata nella slide successiva)
  - **int interseca(Rect r1, Rect r2, Rect * r3)** che, dati due Rect r1 ed r2, se sono almeno parzialmente sovrapposti restituisce 1 ed assegna ad r3 (passato per indirizzo) la loro intersezione, altrimenti (se non vi è intersezione) restituisce 0 e lascia inalterato r3.
- Si scriva infine un programma (main) che, usando le funzioni precedenti, stampi a video il rettangolo di *inviluppo totale* di una serie di rettangoli contenuti in una sequenza di rettangoli rappresentata da un vettore.

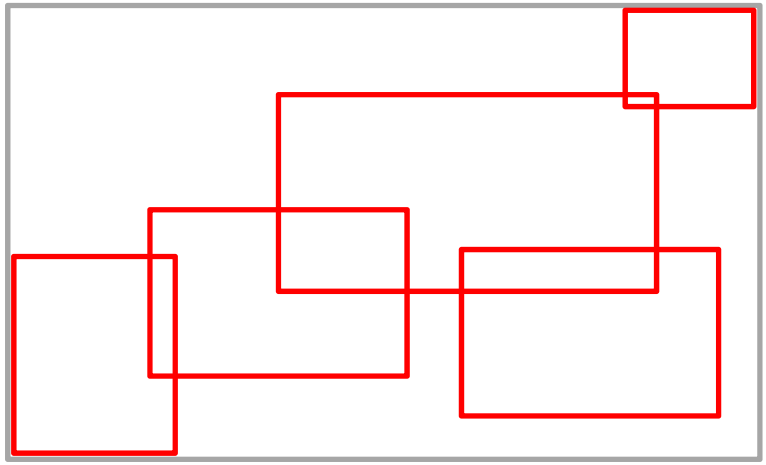
Estensione: si limitino le coordinate a valori interi compresi tra 0;0 e 70;20, e si disegnino i rettangoli tramite caratteri opportuni (+ - | …). Si disegnino i rettangoli del vettore con una "linea singola" e il rettangolo di inviluppo con sequenze di #. Può essere comodo definire a supporto una matrice di caratteri da popolare opportunamente.

Inviluppo di rettangoli

Dati 2 generici rettangoli, il loro rettangolo di inviluppo è il minimo rettangolo che li contiene entrambi. Esempi (rettangoli in rosso, inviluppo in grigio):



La nozione si estende banalmente a N rettangoli:



5. Ancora un po’ di geometria

- Scrivere una funzione **…aligned(…)** che, dati tre punti nel piano, stabilisce se sono allineati
- Scrivere una funzione che, dati due punti nel piano, stampa a video l'equazione della retta che li attraversa, nelle forme

  x = h oppure y = k oppure y = mx + q

  dove x e y sono interpretate come incognite e h,k,m,q sono valori costanti di tipo float
- Scrivere una funzione **float coeffang(…)** che, dati due punti nel piano, restituisce il coefficiente angolare della retta che li attraversa. Come trattare il caso in cui i punti siano verticalmente allineati?
- Scrivere una funzione **…isTrapezio(…)** che, dato un generico poligono di 4 lati, "decide" se è un trapezio (cioè restituisce 1 se è un trapezio, 0 altrimenti), una funzione **…isParallelogramma(…)** che "decide" se un poligono è un parallelogramma, una funzione **…isRombo(…)**, una funzione **…isRettangolo(…)** e una funzione **…isQuadrato(…)**. Si tenti di riusare le funzioni!