Esame di programmazione del 19 giugno 2012

Scrivere in modo leggibile, pena la non correzione. Il tempo a disposizione è di 2 ore e 30 minuti.

Teoria

(i) Data la seguente funzione ricorsiva:

```
int F(nodo* a) {
if(!a) return false;
 if (a->left && a->right) return F(a->right)||F(a->left);
 return !( F(a->left)||F(a->right); )
}
```

Dire cosa calcola F invocata sulle radici dei due alberi raffigurati. Sulla base di questi valori, specificare una appropriata coppia di PRE e POST per la funzione F.

(ii) Considerate il seguente programma. In caso pensiate sia corretto, spiegate cosa calcola e cosa stampa. Se pensate sia sbagliato, spiegate in dettaglio gli errori. In ogni caso, per spiegare le vostre conclusioni, disegnate uno schema dettagliato che mostri la relazione tra le variabili e i puntatori in gioco.

```
int* f(int *&p){int b=3,*x=&b; x=p+1; p++; return x-2; }
main() {int b[]={1,2,3,4},*q=b+2; *f(q)=*q; cout<<b[0]<<b[1]<<b[2]<<b[3];}
```

Programmazione

Introduzione: Il problema da risolvere riguarda il **pattern matching**. Il testo T è un array di interi che va considerato come una matrice R x C e il pattern da cercare in T è un array P di dimP interi. Il match parte dalla prima riga di T e cerca in questa riga P[0], se non lo trova passa alla riga successiva. Se invece trova P[0] in T[i] allora cerca di continuare il match a partire da quella posizione, cioè cerca P[1] in T[i+1] e così via fino a che arrivi alla fine della riga di T oppure il match fallisca. A questo punto si passa alla riga successiva di T, cercando di trovare un match per almeno una parte del resto di P. L'output da produrre è un array Q di R interi tale che Q[k] specifichi quanti elementi di P sono stati trovati nella riga k.

Esempio: Sia T=[0,0,0, 1,1,2, 2,3,0, 0,0,0, 0,4,5]. Gli spazi servono solo a mostrare che T va interpretata come una matrice composta da 5 righe di 3 elementi ciascuna. Quindi R=5 e C=3. Sia P=[1,2,3,4,5,6] e quindi dimP=6. Allora il match non trova nulla sulla prima riga. Riesce a fare match di P[0]=1 sul primo elemento della seconda riga, ma il match fallisce immediatamente dopo perché il secondo elemento della seconda riga (cioèT[3+1]) è 1 che è diverso da P[1]=2. Quindi si passa alla terza riga, dove si riesce a trovare match per P[1..2]=[2,3] in T[6..7]. Visto che P[3]=4 è diverso da T[8]=0, si passa alla quarta riga, dove non si trova alcun match del resto del pattern P[3..5] e quindi si passa alla quinta riga in cui si trova il match di P[3..4]=[4,5] in T[13..14] . L'output che va calcolato è Q=[0,1,2,0,2] che specifica che sulla prima riga non si è trovato nulla, sulla seconda si è trovato P[0], sulla terza P[1..2], sulla quarta nulla, e sulla quinta P[3..4]. Si noti che il match di P non è completo e che la cosa non ha alcuna importanza. Si osservi che è, in linea di principio, possibile anche ottenere Q con tutti 0.

ATTENZIONE: *E' importante capire che, quando si cerca il match di P[k..dimP-1] su una riga di T, si deve cercare il primo match di P[k] e, se lo si trova in T[i], si deve continuare a cercare match di P[k+1..dimP-1] partendo da T[i+1], interrompendosi solo quando si trova un mismatch oppure quando la riga finisce. Se invece il match di P[k] non c'è proprio sulla riga considerata, si passa alla riga successiva.*

Parte iterativa: Si chiede di risolvere il problema proposto con una coppia di funzioni iterative che rispettano i seguenti prototipi: void F(int* T, int R, int C, int * P, int dimP, int *Q) e int matchR(int*riga, int dimr,int*P, int dimP). F deve eseguire l'intero calcolo descritto prima. Il significato dei suoi parametri è spiegato nell'Introduzione. Il significato dei parametri di matchR dovrebbe essere facilmente intuibile. F deve invocare matchR che si deve occupare di fare i calcoli necessari per una riga di T. A voi di capire il significato dell'intero restituito da matchR.

Le PRE e POST di F sono le seguenti:

PRE_F=(T è un array di C*R interi definiti, C ed R > 0, P è un array di dimP interi e dimP>0, Q è un array di R interi tutti a 0)

POST=(Q[0..R-1] contiene il numero di elementi di P che hanno trovato match in ciascuna riga di T, come descritto nell'Introduzione).

Si chiede di scrivere le funzioni iterative F e matchR, di specificare PRE e POST per matchR e di specificare l'invariante del ciclo principale (quello che scorre le righe di T) di F.

Parte ricorsiva: La funzione ricorsiva principale FR, corrisponde a F della parte iterativa, ed è la seguente:

```
//PRE=( i parametri sono tutti definiti, T ha dimT elementi interi, P ne ha dimP e Q ne ha  dimT/C tutti a 0)
void FR(int*T,int dimT, int C, int*Q, int*P, int dimP)
{//caso base: finito T o finito P
 if(!dimT || !dimP) return;
 //c'e' una riga e c'e' pattern
 int x=matchRR(T,C,P,dimP);
 *Q=x;
 FR(T+C,dimT-C,C,Q+1,P+x;dimP-x);
}
//POST=(Q[0..(dimT/C)-1] descrive i match di P[0..dimP-1] nelle dimT/C righe di T come descritto
nell'Introduzione)
```

FR invoca matchRR che sarà la versione ricorsiva di matchR dell'esercizio iterativo. I parametri di FR sono diversi da quelli di F e sono spiegati di seguito. T viene spostata di una riga (C) ad ogni invocazione ricorsiva e il numero dei suoi elementi rimasti è dimT e anch'esso deve diminuire di una riga (C) ad ogni invocazione ricorsiva. Il numero di righe di T è dimT/C, vedi PRE e POST. Anche P, dimP e Q sono trattati nello stesso modo. Questo è coerente con il caso base di FR. Si noti che nella PRE si assume che Q sia inizialmente a 0.

matchRR dovrà invocare almeno 2 funzioni ausiliarie. Una per determinare il match del primo elemento di P nella riga corrente, e, se questo viene trovato, la seconda funzione deve contare di quanto è lungo il match degli elementi successivi di P sul resto della riga.

Si chiede di specificare PRE e POST per ciascuna di queste 3 funzioni. Si chiede inoltre di sviluppare matchRR e le due funzioni ausiliarie che essa deve usare. A scelta per una di queste tre funzioni si delinei la prova di correttezza induttiva.