Compito di Programmazione

26 marzo 2010

Teoria

(1) Dato l'array char X[10][5][10], rispondere ai seguenti due punti:

(i) che tipo ha *(*X-4)+2 e che differenza c'è tra il suo valore e quello di X;

(ii) che tipo ha X[-1] e che differenza c'è tra il suo valore e quello di X;

(2) Dire se è corretto o no il seguente programma, spiegando la propria risposta:

```
int *f(int **p){int b=3,*x=&b; **p=b; x=*p; return x; }
main() {int y=5, b=2,*q=&b; *f(&q)=y*2;}
```

Programmazione ricorsiva: si tratta di un problema di pattern matching di un pattern char P[dim_P] su una lista concatenata L. I nodi di L hanno tipo struct nodo{char info; nodo* next;}; Quello che si vuole è ricercare un match (anche non contiguo) di P su L e, nel caso ci sia, si chiede di togliere da L i nodi che partecipano al match restituendo quello che resta di L e una lista dei nodi staccati. Un esempio dovrebbe chiarire:

Esempio: sia L=a->b->c->a->b e P=[a,a], allora la funzione deve restituire K=a->a e L deve diventare b->c->b. I nodi di K sono il primo ed il quarto nodo di L originale. Se P fosse [a,d,a], non ci sarebbe nessun match su L e quindi la funzione dovrebbe restituire K=0 e L invariata.

Si chiede di rispondere ai seguenti punti:

(i) Scrivere la PRE e POST di una funzione ricorsiva nodo* M(nodo*&L, char*P, int dim_P, bool &ok); che restituisca la lista K dei nodi del match estratti da L con il return (se un match esiste e 0 altrimenti) e nel parametro L produca quello che resta della lista L originale.

(ii) Realizzare il codice della funzione ricorsiva M.

(iii) Dimostrare per induzione che M è corretta rispetto a PRE e POST di (i)

Programmazione iterativa: Si considera lo stesso problema della parte ricorsiva, ma, in questo caso, una parte della soluzione è data e si richiede solo la realizzazione di una funzione ausiliaria. La parte data è la seguente:

```
nodo* MI(nodo*&L, int*P, int dim_P)
 {
  nodo** M=new nodo*[dim_P];
  int n=0;
  nodo* origin=L;
  while(origin && n<dim_P)
    {
     if(origin->info==P[n])
           {
            M[n]=origin;
            n++;
           }
     origin=origin->next;
    }
  nodo* E= 0,*fine;
  if(n==dim_P)
    {
     E=fine=M[0];
     estrae(L,M[0]);
     for(int i=1; i<dim_P;i++)
      {
           estrai(L,M[i]);
            fine->next=M[i];
            fine=M[i];
      }
     fine->next=0;
    }
  delete[] M;
  return E;
}
```

La funzione MI costruisce un array M in cui inserisce puntatori ai nodi di L che partecipano al match. Se all'uscita del ciclo, M è completamente pieno, significa che il match è stato trovato e quindi MI deve estrarre da L i nodi puntati dagli elementi di M (altrimenti non fa nulla). Nella seconda parte di MI viene invocata la funzione ausiliaria iterativa void estrae(nodo*&L, nodo*N), che deve estrarre dalla lista L il nodo puntato da N e restituire in L la lista originale meno il nodo puntato da N. Si può assumere che N punti ad un nodo della lista L.
Si chiede di rispondere ai seguenti punti:

(i) scrivere PRE e POST di estrae.

(ii) realizzare la funzione iterativa estrae.

(iii) corredare il ciclo di estrae di un'asserzione che deve valere all'uscita dal ciclo e di un invariante R e dimostrare che all'uscita dal ciclo effettivamente vale P.