Chi ha superato PI e la prova intermedia deve risolvere gli esercizi PII.2 e PII.3. Chi deve superare l'esame integrato completo deve risolvere gli esercizi PI-PII.1,PI-PII.2,PII.1 e PII.2. Chi ha superato la prova intermedia e non PI deve risolvere tutti gli esercizi tranne PI-PII.2 e PII.1. Chi ha superato PI e non la prova intermedia deve risolvere PI-PII.2, PII.1 e PII.2.

PROGRAMMAZIONE I - PROGRAMMAZIONE II
19/6/2007
Prof. Riccardo Silvestri - Prof.ssa E. Fachini

Cognome:_____

Nome:_____

N. Matricola: _____

Esercizio PI-PII.2

Scrivere una funzione ricorsiva che, presa in input una lista concatenata di interi, fa slittare verso la testa della lista gli elementi applicando il seguente procedimento: se l'etichetta dell'ultimo elemento della lista è minore di quella dell'elemento che lo precede, allora i due elementi si scambiano di posizione, altrimenti si lasciano nelle loro posizioni; quindi, si procede applicando la stessa regola al penultimo elemento (quello ottenuto dopo l'eventuale scambio con l'ultimo elemento) e a risalire sino all'inizio della lista. Si scrivano anche precondizioni e postcondizioni della funzione.

Esempio: la lista di input

L -> 5 -> 8 -> 1 -> 3 -> 10 -> 6 -> 2

viene trasformata in

L -> 1 -> 5 -> 8 -> 2 -> 3 -> 10 -> 6

Soluzione

```
void ordParz(ListaPtr L)
{int temp;
if (!L) return;
 ordParz(L->next);
if (L->next && L->next->elem < L->elem)
      {temp = L->elem;
      L->elem = L->next -> elem;
       L->next -> elem = temp;}
return;}
```

Chi ha superato PI e la prova intermedia deve risolvere gli esercizi PII.2 e PII.3. Chi deve superare l'esame integrato completo deve risolvere gli esercizi PI-PII.1,PI-PII.2,PII.1 e PII.2. Chi ha superato la prova intermedia e non PI deve risolvere tutti gli esercizi tranne PI-PII.2 e PII.1. Chi ha superato PI e non la prova intermedia deve risolvere PI-PII.2, PII.1 e PII.2.

PROGRAMMAZIONE II - 19/6/2007
Prof.ssa E. Fachini

Cognome:_____

Nome:_____

N. Matricola: _____

Esercizio PII.1 (Code e Pile)

Considerate una lista concatenata di interi, e immaginatela divisa in sottoliste massimali di valori strettamente crescenti. Ad esempio, se la lista contiene (nell'ordine) i valori:

L -> 2 -> 3 -> 4 -> 5 -> 5 -> 8 -> 9 -> 7 -> 6 -> 20

le sottoliste crescenti massimali sono le seguenti:

L -> 2 -> 3 -> 4 -> 5 -> | 5 -> 8 -> 9 -> | 7 -> | 6 -> 20

Si definisca una funzione C che modifichi la lista invertendo l'ordine degli elementi nelle sottoliste crescenti massimali. Nell'esempio precedente, la lista modificata dovra' contenere (nell'ordine) i valori:

L -> 5 -> 4 -> 3 -> 2 -> 9 -> 8 -> 5 -> 7 -> 20 -> 6

La funzione deve eseguire una sola scansione degli elementi della lista e deve utilizzare come struttura dati di appoggio una pila o una coda (scegliete la struttura piu' opportuna). Si scrivano anche precondizioni e postcondizioni della funzione.
Si fornisca la specifica (prototipi, precondizioni e postcondizioni e non il codice!!) delle operazioni di gestione dll'ADT che si intende usare (se una coda: accodamento ed estrazione, controllo su coda vuota o piena, o se una pila: push, pop, controllo su pila piena o vuota), in modo che sia possibile prevedere un'implementazione con una lista concatenata o con un vettore.

Soluzione

Chi ha superato PI e la prova intermedia deve risolvere gli esercizi PII.2 e PII.3. Chi deve superare l'esame integrato completo deve risolvere gli esercizi PI-PII.1,PI-PII.2,PII.1 e PII.2. Chi ha superato la prova intermedia e non PI deve risolvere tutti gli esercizi tranne PI-PII.2 e PII.1. Chi ha superato PI e non la prova intermedia deve risolvere PI-PII.2, PII.1 e PII.2.

```
void riordinaCrescenti(ListPtr L)
{
      PilaP P;
 ListPtr inizio =L;
      while (L)
            {
            while (L->next && L->next->elem > L->elem)
                  {
                  Push(P,L->Elem)
                   L = L->next;
                   }
// Si posiziona all'inizio della sottosequenza successiva e
// fa il push dell'ultimo elemento della sottosequenza trovata
            Push(P,L->Elem)
            L=L->next;
// Modifica la sottosequenza appena trovata
            while (!vuota(P))
                  {inizio->elem = pop(P);
                   inizio = inizio->next;
                  }
      }
}
```

Chi ha superato PI e la prova intermedia deve risolvere gli esercizi PII.2 e PII.3. Chi deve superare l'esame integrato completo deve risolvere gli esercizi PI-PII.1,PI-PII.2,PII.1 e PII.2. Chi ha superato la prova intermedia e non PI deve risolvere tutti gli esercizi tranne PI-PII.2 e PII.1. Chi ha superato PI e non la prova intermedia deve risolvere PI-PII.2, PII.1 e PII.2.

PROGRAMMAZIONE II - 19/6/2007
Prof.ssa E. Fachini

Cognome:_____

Nome:_____

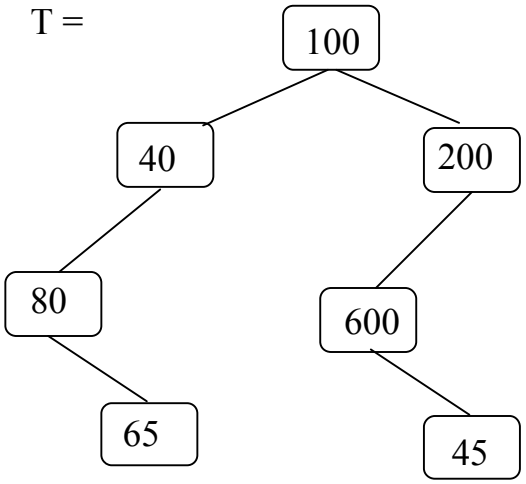
N. Matricola: _____

Esercizio PII.2 (Ricorsione su alberi)

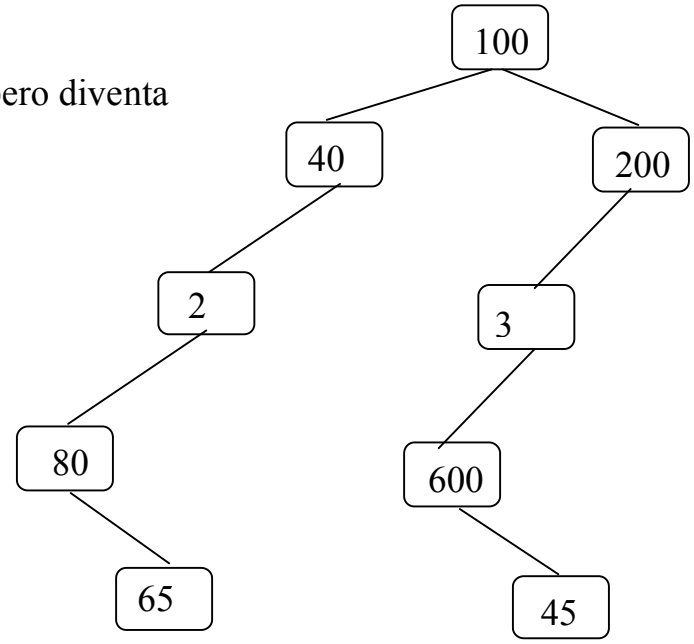
Sia T un albero binario contenente valori interi. Modificare T come segue: per ogni nodo u con valore val(u) che ha un solo figlio v con valore val(v) tale che val(v) e' multiplo di val(u), inserire un nuovo nodo z con valore val(z) = val(v) / val(u). Se v e' figlio sinistro di u, allora z deve essere figlio sinistro di u e v deve diventare figlio sinistro di z. Se v e' figlio destro di u, allora z deve essere figlio destro di u e v deve diventare figlio destro di z. Si scrivano anche precondizioni e postcondizioni della funzione.

Esempio

su input
T =



l'albero diventa



Chi ha superato PI e la prova intermedia deve risolvere gli esercizi PII.2 e PII.3. Chi deve superare l'esame integrato completo deve risolvere gli esercizi PI-PII.1,PI-PII.2,PII.1 e PII.2. Chi ha superato la prova intermedia e non PI deve risolvere tutti gli esercizi tranne PI-PII.2 e PII.1. Chi ha superato PI e non la prova intermedia deve risolvere PI-PII.2, PII.1 e PII.2.

Soluzione

```
void aggiungiMultipli(TreePtr T) {
	if (!T) return;

	if (T->lPtr && !(T->rPtr) && T->lPtr->elem % T->elem == 0) {
			TreePtr temp = malloc(sizeof(struct nodoalbero));
			assert(temp);
			temp->elem = T->lPtr->elem / T->elem;
			temp->lPtr = T->lPtr;
			temp->rPtr = NULL;
			T->lPtr = temp;
			aggiungiMultipli(temp->lPtr);
	}
	else if (T->rPtr && !(T->lPtr) && T->rPtr->elem % T->elem == 0) {
			TreePtr temp = malloc(sizeof(struct nodoalbero));
			assert(temp);
			temp->elem = T->rPtr->elem / T->elem;
			temp->rPtr = T->rPtr;
			temp->lPtr = NULL;
			T->rPtr = temp;
			aggiungiMultipli(temp->rPtr);
	}
	else {
		aggiungiMultipli(T->lPtr);
		aggiungiMultipli(T->rPtr);
	}
}
```

Chi ha superato PI e la prova intermedia deve risolvere gli esercizi PII.2 e PII.3. Chi deve superare l'esame integrato completo deve risolvere gli esercizi PI-PII.1,PI-PII.2,PII.1 e PII.2. Chi ha superato la prova intermedia e non PI deve risolvere tutti gli esercizi tranne PI-PII.2 e PII.1. Chi ha superato PI e non la prova intermedia deve risolvere PI-PII.2, PII.1 e PII.2.

PROGRAMMAZIONE II - 22/2/2007
prof.ssa E. Fachini

Cognome:_____

Nome:_____

N. Matricola: _____

Esercizio PII.3 (Testing)

Si individuino opportuni dati di test per la funzione sottostante evidenziando per quali dati si è seguito l'approccio a scatola nera e per quali quello a scatola trasparente. Si dia una breve motivazione della scelta.

```
int restIesimo ( TreePtr t, int *i)
/*prec: t è un ABR && T!= NULL && 1≤i≤ numNodi(t)
postc: restituisce l'i-simo valore nell'albero */
{int temp ;
if (t->left) temp = restIesimo(t->left,i);
if (*i== 1) {  *i = 0; temp= t->elem; }
if (*i !=0)  /* per uscire una volta trovato l'elemento */
{(*i)--;
if (t->right) temp =  restIesimo(t -> right,i);}
return temp; }
```

A scatola nera:
un albero con un solo nodo e i=1.
I due alberi con due nodi e i=1,2.
I due alberi con 3 nodi con figlio sinistro della radice che ha figlio destro e viceversa e sempre i=1,2,3.
L'albero con 3 nodi, in cui la radice ha due figli e i =1,2,3.
A scatola trasparente:
I casi di nessuna chiamata, una sola chiamata a sinistra e una sola a destra oppure una a sinistra e una a destra sono già contemplati a scatola nera.
Prendiamo alberi con 3 nodi e in ogni caso tutti i possibili valori di i:

Chi ha superato PI e la prova intermedia deve risolvere gli esercizi PII.2 e PII.3. Chi deve superare l'esame integrato completo deve risolvere gli esercizi PI-PII.1,PI-PII.2,PII.1 e PII.2. Chi ha superato la prova intermedia e non PI deve risolvere tutti gli esercizi tranne PI-PII.2 e PII.1. Chi ha superato PI e non la prova intermedia deve risolvere PI-PII.2, PII.1 e PII.2.
Quello tutto sbilanciato a sinistra, tutto sbilanciato a destra, così copro i casi di due chiamate a sinistra e due chiamate a destra, infine con l'albero completo di altezza 2 copro i casi di due chiamate a destra e due chiamate a sinistra.