Esercitazioni di Prog. II (funzioni su insiemi)

Chiara Petrioli

Esercizi per la manipolazione di insiemi (rappresentati tramite liste)

Insiemi.c

```
/*Questo file include i prototipi e le definizioni di specifiche funzioni e procedure per manipolare insiemi verifica che l'insieme non sia vuoto;
        calcolo della cardinalita' dell'insieme; inserimento di un nuovo elemento; cancellazione di un elemento dell'insieme; unione di insiemi
        rappresentati con liste non ordinate (iterativa);  intersezione di insiemi rappresentati con liste non ordinate (iterativa e ricorsiva);
        differenza di insiemi rappresentati con liste non ordinate (iterativa). LE PROCEDURE CHE COMINCIANO CON R indicano le versioni
        ricorsive*/
#include <stdio.h>
#include <stdlib.h>

struct elemento_insieme{
int elem;
struct elemento_insieme *next;
};

typedef struct elemento_insieme     ELEM;
typedef ELEM     *I_PTR;

int isempty(I_PTR)
int Rcardinalita(I_PTR)
.....
void unione (I_PTR *, I_PTR, I_PTR);
void intersezione (I_PTR *, I_PTR, I_PTR);
I_PTR Rintersezione (I_PTR,I_PTR);
void differenza (I_PTR *, I_PTR, I_PTR);

main()
…..
```

Esercizio 1

```
/*verifica se l'insieme rappresentato da una lista non ordinata e' vuoto
Pre: L insieme (ovvero dati comunque e1 e2 elementi dell'insieme e1!=e2) di interi
 Post: restituisce 1 se l'insieme e' vuoto, 0 altrimenti
 */

int isempty(I_PTR L)
{
return (L==NULL);
}
```

Si scriva una funzione che, dato un insieme Verifichi se sia vuoto

```
/* calcola il numero di elementi di un insieme rappresentato da una lista non ordinata
 * Pre: L insieme (ovvero dati comunque e1 e2 elementi dell'insieme e1!=e2) di interi
 * Post: restituisce il numero di elementi dell'insieme
 * */

int Rcardinalita(I_PTR L)
{
      if (L!=NULL)
            return(1+Rcardinalita(L->next));
      else
            return 0;
}
```

Si scriva una funzione che, dato un insieme Ne calcoli la cardinalita'

Esercizio 2

```
/* restituisce un puntatore ad una nuovo insieme (rappresentato con una lista non rodinata)
* i cui elementi sono l'unione di due insiemi passati come parametro
* , ovvero un elemento e' nell'insieme unione se e solo se il valore dell'elemento compare
* in almeno uno dei due insiemi.
* Pre: L1 ed L2 rappresentano insiemi (ovvero dati comunque e1 e2 elementi dell'insieme
* e1!=e2)di valori interi
* Post: PTRLISTA punta ad un insiemi i cui valori contengono L1 U  L2
*   */


void unione (I_PTR *PTRINSIEME, I_PTR L1, I_PTR L2)
{
      I_PTR curr, temp;
      temp=NULL;
      PTRINSIEME=&temp;

      curr = L1;
      while (curr!=NULL)
      {
            if (!(Rfindlista(*PTRINSIEME,curr->elem)))
                  insertHEADlista(PTRINSIEME,curr->elem);
            curr = curr->next;
      }
      curr = L2;
      while (curr!=NULL)
      {
            if (!(Rfindlista(*PTRINSIEME,curr->elem)))
                  insertHEADlista(PTRINSIEME,curr->elem);
            curr = curr->next;
      }

}
```

Si scriva una funzione
Procedura che
Dati due insiemi
Costruisca un insieme
Che ne contenga l'unione

Esercizio 3 –intersezione di insiemi

```
/* restituisce un puntatore ad un insieme di interi rappresentato con una lista non
 * ordinata i cui elementi sono l'intersezione di due insiemi passati
 * per parametro, ovvero un elemento e' nell'insieme intersezione se e solo se il
    valore  dell'elemento compare in entrambe gli insiemi.
 * Pre: L1 ed L2 rappresentano insiemi(ovvero dati comunque e1 e2 elementi
    dell'insieme e1!=e2) di valori interi
 * Post: PTRLISTA punta ad un insieme i cui valori contengono L1 intersezione L2
 *  */

void intersezione (I_PTR *PTRINSIEME, I_PTR L1, I_PTR L2)
{
    I_PTR curr, temp;
    temp=NULL;
    PTRINSIEME=&temp;

    curr = L1;
    while (curr!=NULL)
    {
        if ((Rfindlista(L2,curr->elem))&&(!(Rfindlista(*PTRINSIEME,curr->elem))))
                insertHEADlista(PTRINSIEME,curr->elem);
        curr = curr->next;
    }
}
```

Si scriva una procedura che dati due insiemi Costruisca l'insieme intersezione

Esercizio 4-differenza di insiemi

```
/*restituisce un puntatore ad un insieme di interi che sia  la differenza di due
      insiemi rappresentati dalle liste non ordinate L1 ed L2 (ovvero un elemento e'
      contenuto nell'insieme differenza L1-L2 se e solo se appartiene al primo
      insieme ma non al secondo.
Pre: L1 ed L2 rappresentano insiemi (ovvero dati comunque e1 e2 elementi
      dell'insieme e1!=e2) di interi
 Post: PTRLISTA punta all'insieme  L1-L2 */

void differenza (I_PTR *PTRINSIEME, I_PTR L1,
 I_PTR L2)
{
      I_PTR curr, temp;
      temp=NULL;
      PTRINSIEME=&temp;
      curr = L1;

      while (curr!=NULL)
      {
            if ((!(Rfindlista(L2,curr->elem)))&&(!Rfindlista(*PTRINSIEME,curr->elem)))
                  insertHEADlista(PTRINSIEME,curr->elem);
            curr = curr->next;
      }
}
```

Si scriva una procedura
Che dati due insiemi
Costruisca l'insieme
Differenza (ovvero l'insieme
In cui compaiono gli
Elementi del primo insieme
Che non sono presenti
Nel secondo)

-Cosa succede se per errore L1 o L2 sono collezioni?
-Sapete scrivere le versioni ricorsive di queste procedure?

Esercizio 5-intersezione tra insiemi
(realizzati con liste ordinate)

```
/* Pre:  le liste L1 e L2 sono ordinate in ordine non decrescente e non  contengono elementi duplicati */
/* Post: restituisce la lista intersezione */
I_PTR intersezioneOrdinate(I_PTR L1, I_PTR L2) {
        I_PTR temp;

        if ((L1==NULL) || (L2==NULL)) return NULL;
        else if (L1->elem < L2->elem)
            return(intersezioneOrdinate(L1->next,L2));
        else if (L1->elem >L2->elem)
            return(intersezioneOrdinate(L1, L2->next));
        else
        {
                temp = malloc(sizeof(struct elemento_insieme));
                temp->elem = L1->elem;
                temp->next = intersezioneOrdinate(L1->next,L2->next);
                return temp;
         }
}
```

Si scriva una procedura Che dati due insiemi Costruisca l'insieme intersezione

Esercizio 6-differenza tra insiemi
(realizzati con liste ordinate)

```
/* Pre:  le liste L1 e L2 sono ordinate in ordine non decrescente e non  contengono elementi duplicati */
/* Post: restituisce la lista intersezione */
I_PTR differenzaOrdinate(I_PTR L1, I_PTR L2) {
        I_PTR L;

        if (L1==NULL) return NULL;
        else if ((L2==NULL)|| (L1->elem < L2->elem))
        {
                L = malloc(sizeof(struct elemento_insieme));
                if (!L) {
                        printf("Allocazione di memoria fallita
                        nella creazione della lista intersezione");
                        return NULL;
                }
                L->elem = L1->elem;
                L->next = differenzaOrdinate(L1->next,L2);
                return L;
        }
        else if (L1->elem > L2->elem)
                return differenzaOrdinate(L1,L2->next);
        else
                return differenzaOrdinate(L1->next,L2->next);
}
```

Si scriva una procedura Che dati due insiemi Costruisca l'insieme differenza