ESERCIZI RISOLTI SULLE LISTE

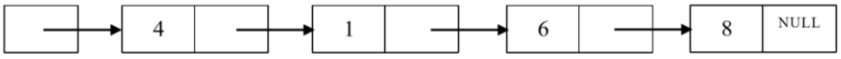
A cura del prof. Marco Cococcioni

Esercizio 1 (04-09-2015)

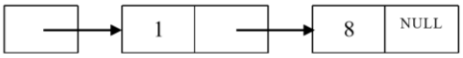
Sia data la struttura seguente

```
struct elem {int info; elem* pun;};
```

Scrivere una funzione che, data una lista di elementi di tipo elem, elimina tutti gli elementi in posizione dispari. Assumere che il primo elemento abbia posizione 1.
Per esempio, se la funzione viene chiamata con la lista seguente:



la lista viene modificata come segue:

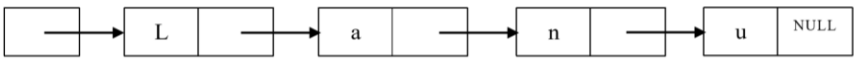


Esercizio 2 (23-07-2015)

Sia data la struttura seguente

```
struct elem {char lettera; elem* pun;};
```

Scrivere una funzione che, data una stringa di lettere (maiuscole e/o minuscole) passata come argomento alla funzione, restituisce una lista di strutture di tipo `elem`, dove il campo `lettera` di ciascuna struttura contiene una lettera della stringa. La lista deve essere ordinata crescente. Per esempio, se la stringa passata alla funzione è "Luna", la funzione restituisce la seguente lista:



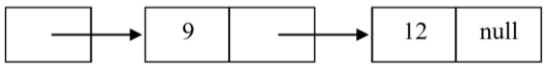
Esercizio 3 (18-02-2015)

Sia data la struttura elem seguente:

```
struct elem {int info;  elem * pun; };
```

Scrivere una funzione che, data una lista L di elementi di tipo `elem`, restituisca una nuova lista ordinata per valori crescenti del campo `info` che contiene tutti gli elementi della lista L che sono multipli di 3. La lista L non deve essere modificata.
Per esempio, se la funzione viene chiamata con la lista L seguente:

la funzione restituisce la lista seguente:

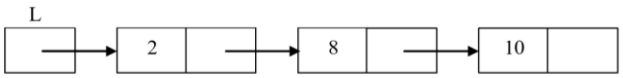


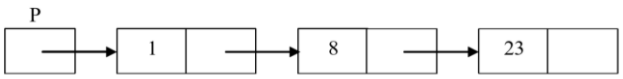
Esercizio 4 (29-01-2015)

Sia data la struttura seguente:

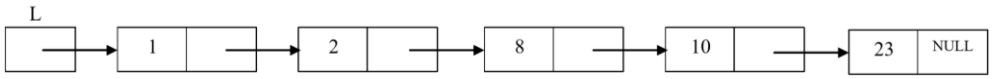
```
struct elem {int info; elem* pun;};
```

Scrivere una funzione che, date due liste di elementi di tipo elem ordinate per valori crescenti del campo info, inserisce nella prima lista tutti gli elementi della seconda, in modo tale che non siano presenti duplicati e che la lista sia mantenuta ordinata.
Nell'esempio seguente, siano L e P i puntatori alla testa della prima e della seconda lista, rispettivamente.





Dopo la chiamata alla funzione, la lista L è la seguente:

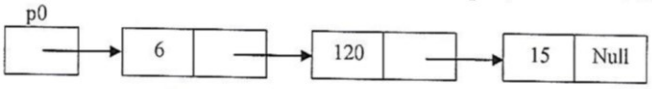


Esercizio 5 (17-09-2014)

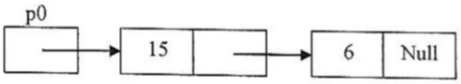
Sia data la struttura seguente

```
struct elem{int info; elem* pun;};
```

Scrivere una funzione che, data una lista p0 di strutture di tipo elem, e due inetri n e m, elimini dalla lista tutti gli elementi con campo informazione compreso fra n e m (n e m inclusi). La funzione restituisce il numero di elementi eliminati. Per esempio, data la lista



se n=100 e m=130, la funzione restituisce 1 e modifica la lista come segue:

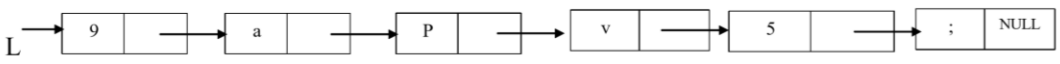


Esercizio 6 (02-07-2014)

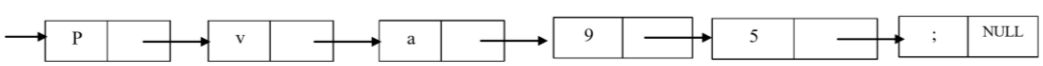
Sia data la struttura seguente:

```
struct elem {char info; elem* pun;};
```

Scrivere una funzione che prende come argomento una lista L di elementi di tipo elem e restituisce una nuova lista che contiene tutti gli elementi di L organizzati in tre gruppi come segue:
il primo gruppo contiene gli elementi il cui campo informazione è una lettera maiuscola o minuscola;
il secondo gruppo contiene gli elementi il cui campo informazione è una cifra;
il terzo gruppo contiene tutti i rimanenti elementi.
All'interno dello stesso gruppo non è importante l'ordine degli elementi.
Per esempio, se la funzione viene chiamata con la lista L seguente:



la funzione potrebbe restituire una nuova lista come segue:

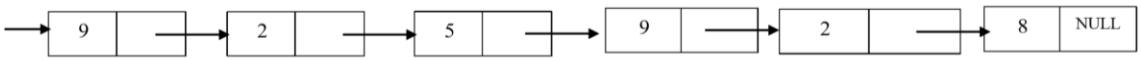


Esercizio 7 (11-06-2014)

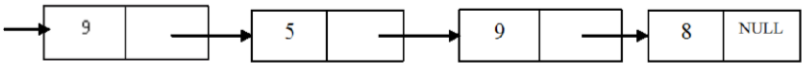
Sia data la struttura seguente:

```
struct elem {int info; elem* pun;};
```

Scrivere una funzione che prende come argomento una lista non ordinata di elementi di tipo `elem` e modifica la lista eliminando tutti gli elementi che hanno campo informazione uguale al minimo numero intero presente nella lista.
Per esempio, se la funzione viene chiamata con la lista seguente:



la funzione modifica la lista come segue:

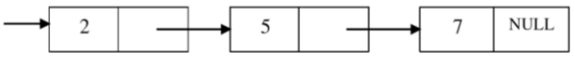


Esercizio 8 (19-02-2014)

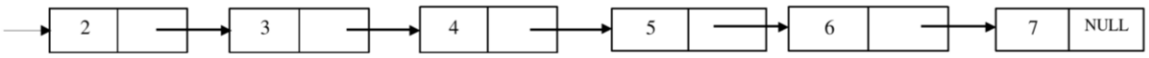
Sia data la struttura seguente:

```
struct elem {int info; elem* pun;};
```

Scrivere una funzione che prende come argomento una lista ordinata e senza duplicati di elementi di tipo `elem` e la completa inserendo fra ogni coppia di elementi tutti gli eventuali elementi mancanti.
Per esempio, se la funzione viene chiamata con la lista seguente:



la funzione modifica la lista come segue:

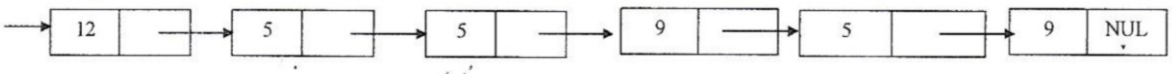


Esercizio 9 (30-01-2014)

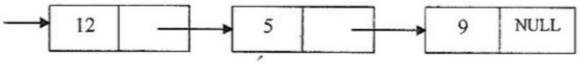
Sia data la struttura seguente:

```
struct elem {int info; elem* pun;};
```

Scrivere una funzione che prende come argomento una lista di elementi di tipo `elem` e modifica la lista eliminando tutti i duplicati.
Per esempio, se la funzione viene chiamata con la lista seguente:



la funzione modifica la lista come segue:



SOLUZIONI

Soluzione Esercizio 1

```
void elimina(elem * & p0){
    if (p0 == NULL)
         return;
    bool dispari = true;
    elem * r, * s;
    r = p0;
    while (r != NULL){
      if (!dispari){
        s = r;
        r = r -> pun;
        dispari = true;
      } else {
        if (r == p0){
          p0 = p0 -> pun;
          delete r;
          r = p0;
        } else {
          s -> pun = r -> pun;
          delete r;
          r = s -> pun;
        }
        dispari = false;
      }
    } // fine while }
```

Soluzione Esercizio 2

```
elem * crealista(const char * v){
    int i = 0;
    elem * p, * q, * r;
    elem * testa = NULL;
    while (v[i] != '\0'){
       r = new elem;
       r -> lettera = v[i];
       for (p = testa;  p != NULL && p -> lettera <= v[i];  p = p -> pun)
          q = p;
       r -> pun = p;
       if (p == testa)
          testa = r;
       else
          q -> pun = r;
       i++;
    } // fine while
    return testa;
}
```

Soluzione Esercizio 3

```
elem * crea_lista(elem * L){
    elem * testa = NULL;
    elem * q, * r, * s, * t;
    for (q = L; q != NULL; q = q -> pun){
       if ((q -> info % 3) == 0){
          t = new elem;
          t -> info = q -> info;
          for (r = testa, r != NULL && r -> info < q -> info; r = r -> pun)
              s = r;
          t -> pun = r;
          if (r == testa)
             testa = t;
          else
             s -> pun = t;
       }  // fine if
    }  // fine for
    return testa;
}
```

Soluzione Esercizio 4

```
void insordinato(elem * & p0, int k){
  elem * r, * s;
  for (s = p0; s != NULL && s -> info < k; s = s -> pun)
    r = s;
  if (s != NULL && s -> info == k)
    return;
  elem = t = new elem;
  t -> info = k;
  if (s == p0){
     t -> pun = p0;
     p0 = t;
  }
  else{
     r -> pun = t;
     t -> pun = s;
  }
}

void modificaLista(elem * & L, elem * P){
  for (elem * q = P; q != NULL; q = q -> pun)
    insordinato(L, q -> info);
}
```

Soluzione Esercizio 5

```
void elimina(elem * & p0, int n, int m){
    if (p0 == NULL)
      return;

    int min = (n <= m) ? n : m;
    int max = (n >= m) ? n : m;

    elem * p = p0;
    elem * q;
    while (p != NULL){
      if ((p -> info >= min) && (p -> info <= max)){
        if (p == p0){
           p0 = p -> pun;
           delete p;
           p = p0;
        }else{
           q -> pun = p -> pun;
           delete p;
           p = q -> pun;
        }
      } // fine cancellazione
      else{
          q=p;
          p = p->pun;
      }
   }
}
```

Soluzione Esercizio 6

```
bool lettera(char c) {
  if (((c >= 'A') && (c <= 'Z')) || ((c >= 'a') && (c <= 'z')))
    return true;
  return false;
}

bool cifra(char c) {
  if ((c >= '0') && (c <= '9'))
    return true;
  return false;
}

elem * creaLista(elem * testa) {
  if (testa == NULL) return NULL;
  elem * nuova = new elem;
  nuova -> info = testa -> info;
  nuova -> pun = NULL;

  elem * prec, * r, * q, * n;

  for (q = testa -> pun; q != NULL; q = q -> pun) {
    n = new elem;
    n -> info = q -> info;
    n -> pun = NULL;

    if (lettera(n -> info)) /* inserimento in testa */ {
      n -> pun = nuova;
      nuova = n;
    } else {
      if (cifra(n -> info)) /* inserimento dopo le lettere */ {
        for (r = nuova;
          (r != NULL) && lettera(r -> info); r = r -> pun) prec = r;
        if (r == nuova) {
          n -> pun = nuova;
          nuova = n;
        } else {
          prec -> pun = n;
          n -> pun = r;
        }
      } else /* inserimento in fondo */ {
        for (r = nuova;
          (r -> pun != NULL); r = r -> pun);
        r -> pun = n;
      }
    }
  }
  return nuova;
}
```

Soluzione Esercizio 7

```
void eliminaMinimo(elem * & testa){
  if (testa == NULL)
    return;
  int min = testa -> info;
  elem * q;
  for (q = testa -> pun; q != NULL; q = q -> pun)
    if (q -> info < min)
      min = q -> info;

  q = testa;
  while ((q != NULL) && (q -> info == min)){
    testa = testa -> pun;
    delte q;
    q = testa;
  }
  if (q == NULL)
    return;

  elem * prec = q;
  q = q -> pun;
  while (q != NULL){
    if (q -> info == min){
      prec -> pun = q -> pun;
      delete q;
      q = prec -> pun;
    }else{
      p = q;
      q = q -> pun;
    }
  }
  return;
}
```

Soluzione Esercizio 8

```
void f(elem * p0){
  elem * q, * p;
  if (p0 == NULL)
    return;
  q = p0;
  p = p0 -> pun;
  while (p != NULL){
    for (int inizio = (q -> info) + 1; inizio < p -> info; inizio++){
      q -> pun = new elem;
      q = q -> pun;
      q -> info = inizio;
    }
    q -> pun = p;
    q = p;
    p = p -> pun;
  }
}
```

Soluzione Esercizio 9

```
void elimina_dup(elem * testa){
  int numero;
  elem * prec;
  for (elem * q = testa; q != NULL; q = q -> pun){
    numero = q -> info;
    prec = q;
    for (elem * p = q -> pun; p != NULL; p = p -> pun){
      if (p -> info == numero){
        prec -> pun = p -> pun;
        delete p;
        p = prec -> pun;
      }
      else
         prec = p;
    } // fine for interno
  } // fine for esterno
}
```