1)

```
bool trova(elem* p, int x){
   //casi base
   if (p==NULL) return false;
   if (p->info==x) return true;
   //ricorsione
   return trova(p->pun, x);
}
```

2)

```
//restituisce TRUE se elimina l'elemento, FALSE se non lo trova
//elimina un solo elemento, quindi la funzione va chiamata iterativamente fin
  quando non restituisce FALSE
bool elimina_elem(elem*& p0, int x){
   elem* p=p0;
   elem* q=p0;
   while(q!=NULL && q->info!=x){
         p=q;
         q=q->pun;
   }
   //q punta l'elemento da eliminare
   //caso 1: elemento non trovato
   if(q==NULL){return false;}
   //caso 2: eliminazione dell'elemento puntato da q, facendo attenzione al
   caso di eliminazione in testa
   if (q==p0) p0 = p0->pun;
   else p->pun=q->pun;
   delete q;
   return true;
}

//modifica la lista p1, mentre p2 non viene modificata
void elimina_lista(elem*& p1, elem* p2){
   while(p1!=NULL && p2!=NULL){
         while(elimina_elem(p1, p2->info));
         p2=p2->pun;
   }
}
```

3)

```
//decrementa di 5 l'elemento in posizione i
//n e' la dimensione del vettore
//restituisce true se il decremento viene effettuato con successo, false
  altrimenti
bool decrementa(int vett[], int n, int k){
   if (k<n && k>=0)
   {
      vett[k]-=5;
      //ri-ordina
      for(int i=0; i<k; i++)
      {
            if (vett[i]>vett[k])
            {
               int temp = vett[k];
               for (int j=k; j>i; j--)
                  vett[j] = vett[j-1];
               vett[i] = temp;
               return true;
```

```
                }
            }
        }
        return false;
    }
```

4) $(1111101110101000011001)_2$
$(95FE)_{16}$

5)

4
10
5
B
D
4
10
5
10
E

6)

(1)
a
Ecc. da f1: a
Ecc. da main: ?

(2)
b
Ecc. da f: b
b

(3)
c
Ecc. da main: ?