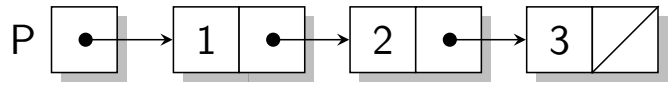
Exercice 1

$P \leftarrow$ créerListe(1, créerListe(2, créerListe(3, NULL))) ;

Exercice 1

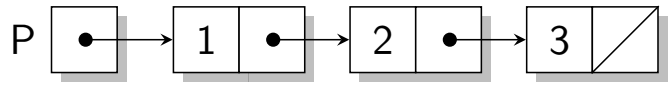
P $\leftarrow$ créerListe(1, créerListe(2, créerListe(3, NULL))) ;



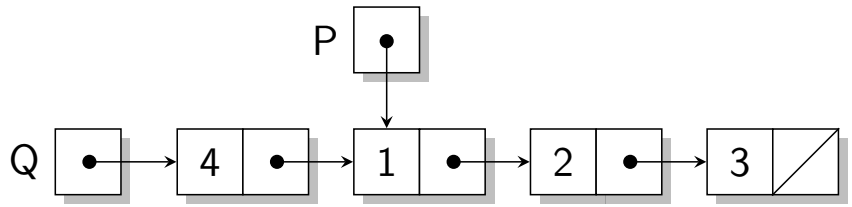
Q $\leftarrow$ créerListe(4,P) ;

Exercice 1

P ← créerListe(1, créerListe(2, créerListe(3, NULL))) ;



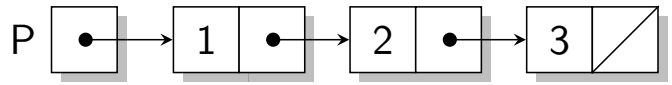
Q ← créerListe(4,P) ;



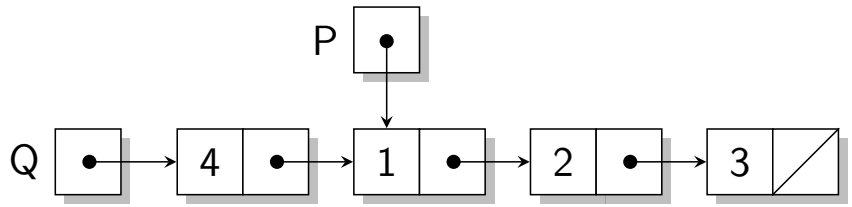
(P↑ succ) ↑ info ← 6 ;

Exercice 1

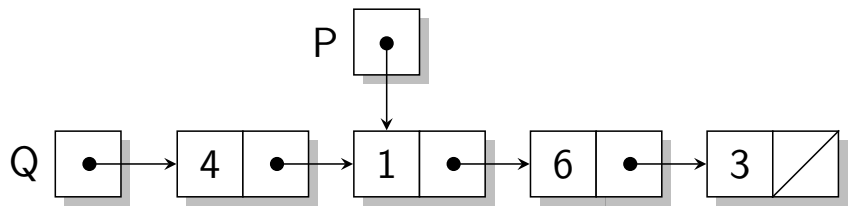
P $\leftarrow$ créerListe(1, créerListe(2, créerListe(3, NULL))) ;



Q $\leftarrow$ créerListe(4,P) ;



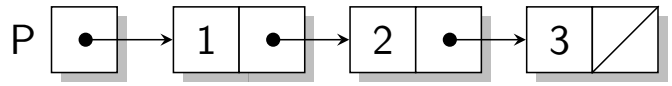
(P$\uparrow$ succ) $\uparrow$ info $\leftarrow$ 6 ;



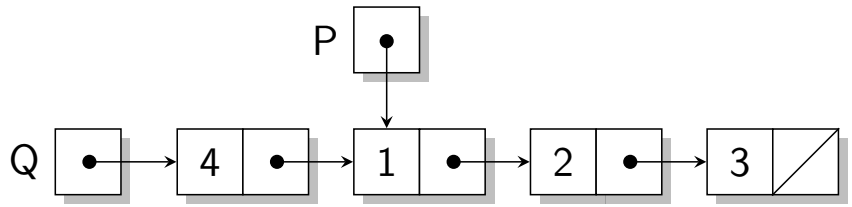
R$\leftarrow$ créerListe(5, (P$\uparrow$succ)$\uparrow$succ) ;

Exercice 1

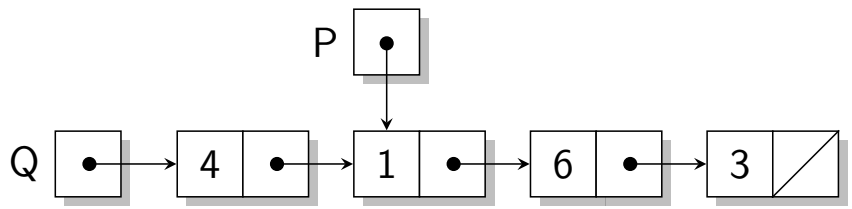
P ← créerListe(1, créerListe(2, créerListe(3, NULL))) ;



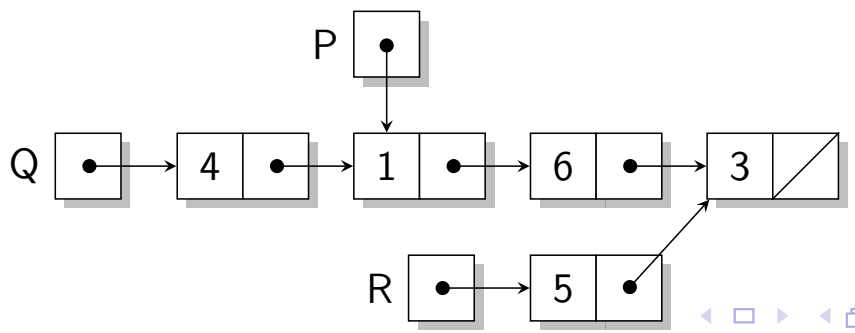
Q ← créerListe(4,P) ;

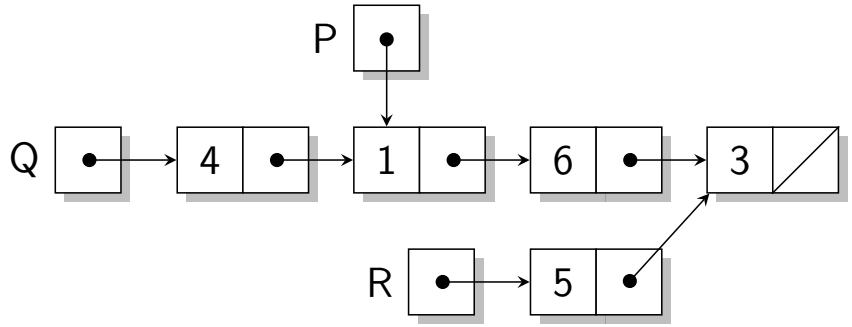


(P↑ succ) ↑ info ← 6 ;

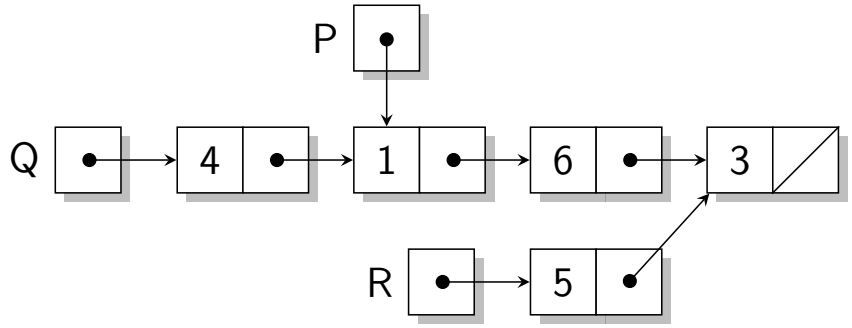


R← créerListe(5, (P↑succ)↑succ) ;

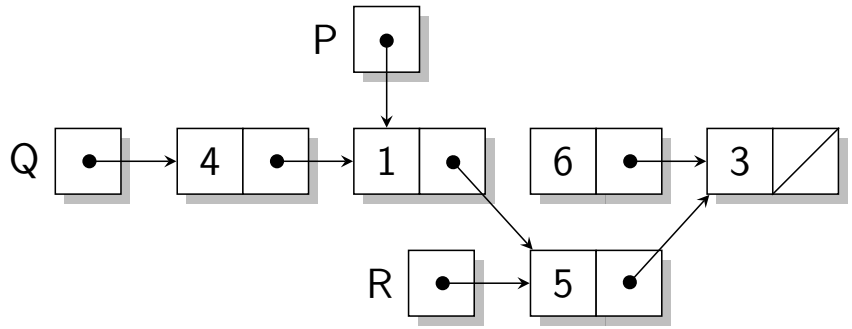




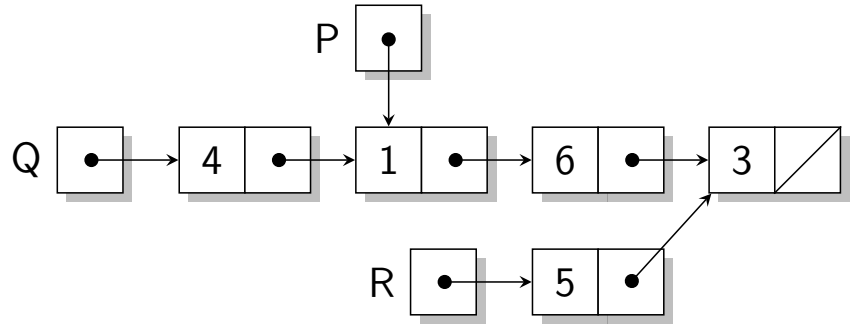
(P↑succ) ← R ;



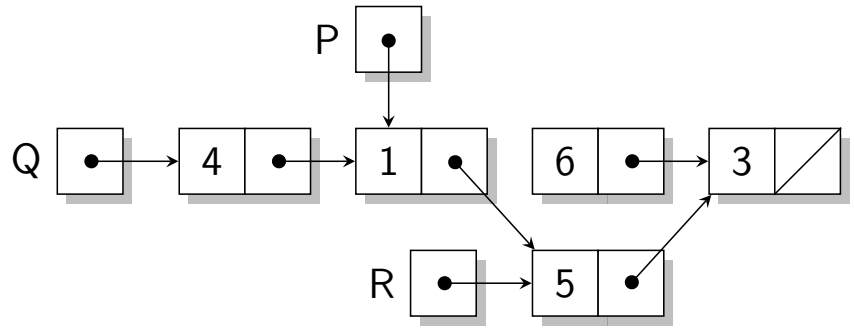
(P↑succ) ← R ;



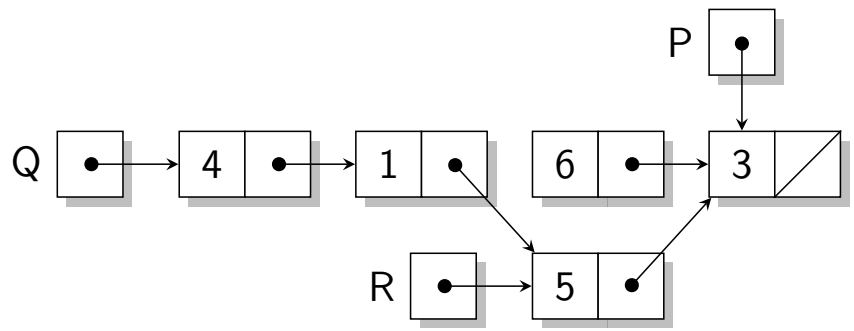
P← R ↑ succ ;

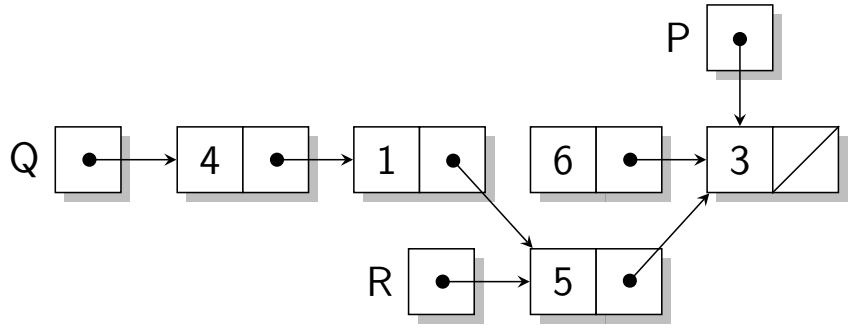


(P↑succ) ← R ;

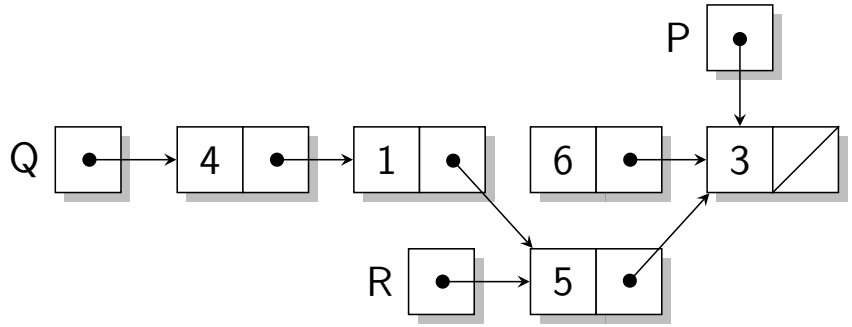


P← R ↑ succ ;

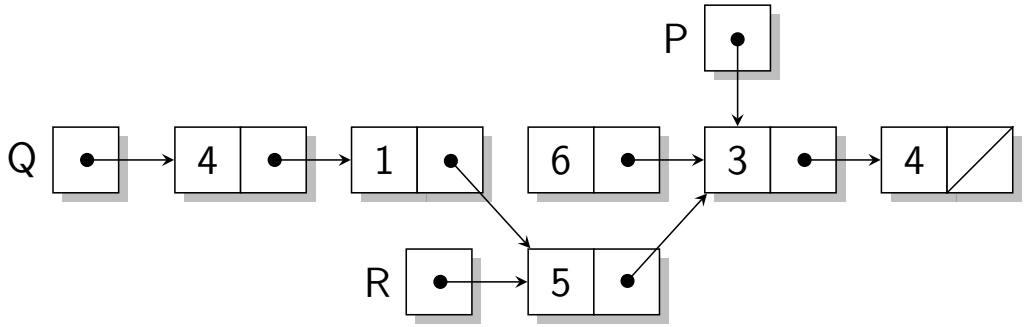




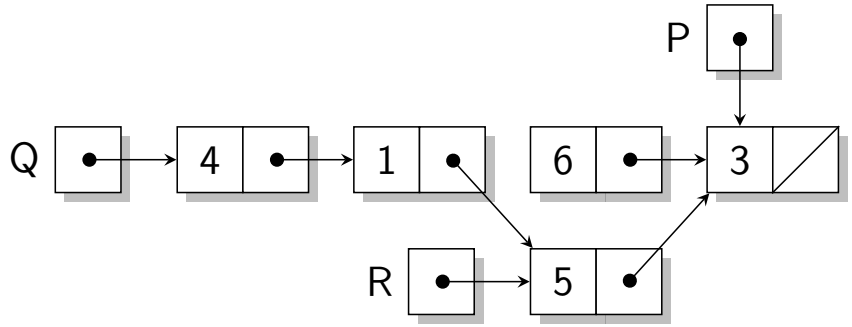
insérerFin(R,Q↑info) ;



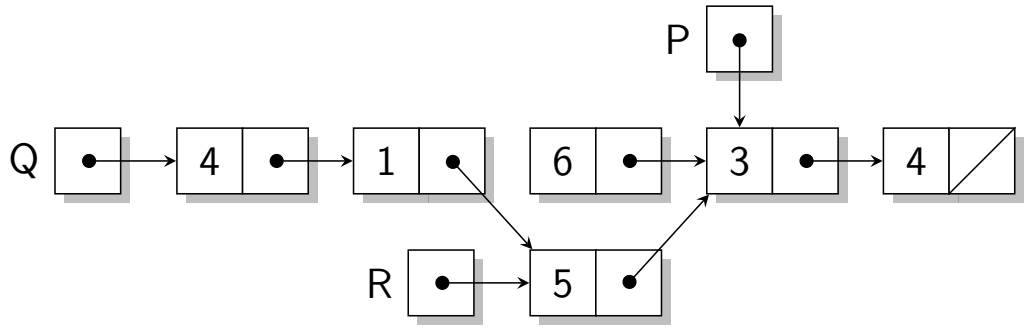
insérerFin(R,Q↑info) ;



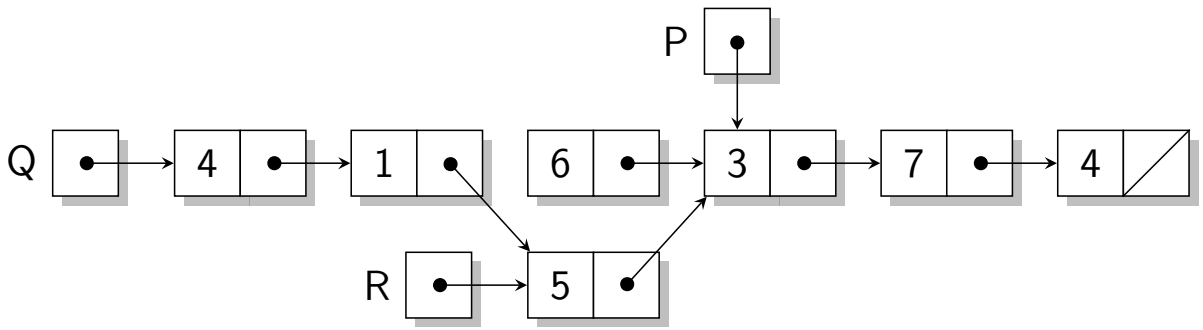
insérerAprès(Q,P,7) ;

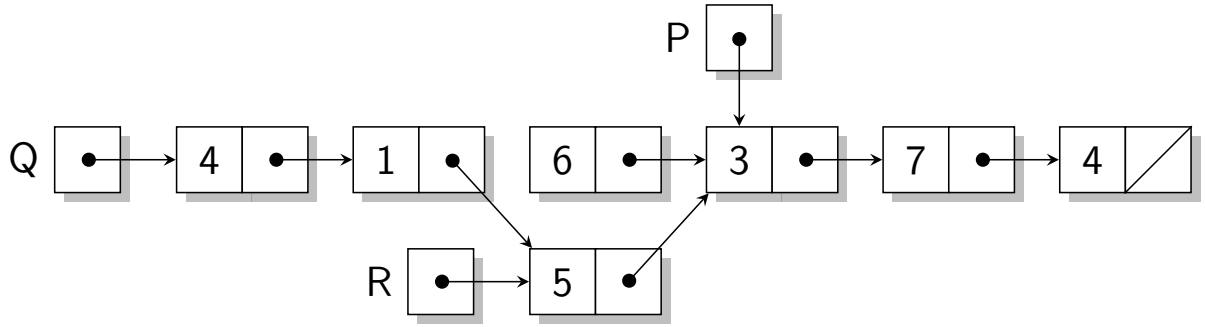


insérerFin(R,Q↑info) ;

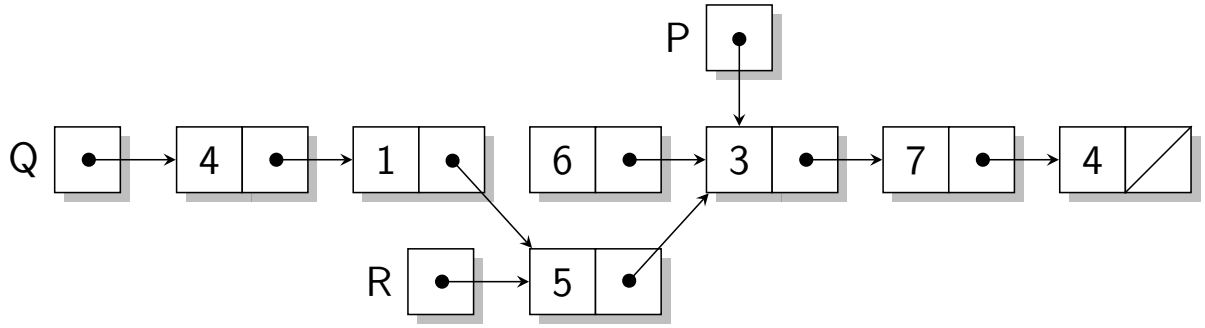


insérerAprès(Q,P,7) ;

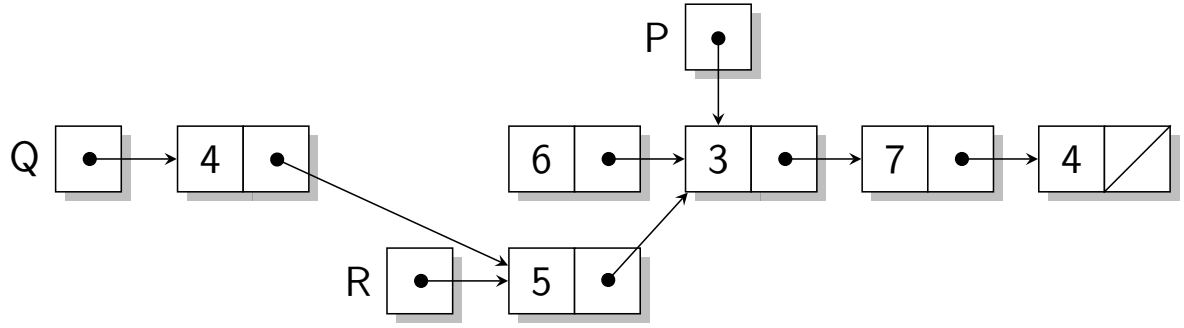




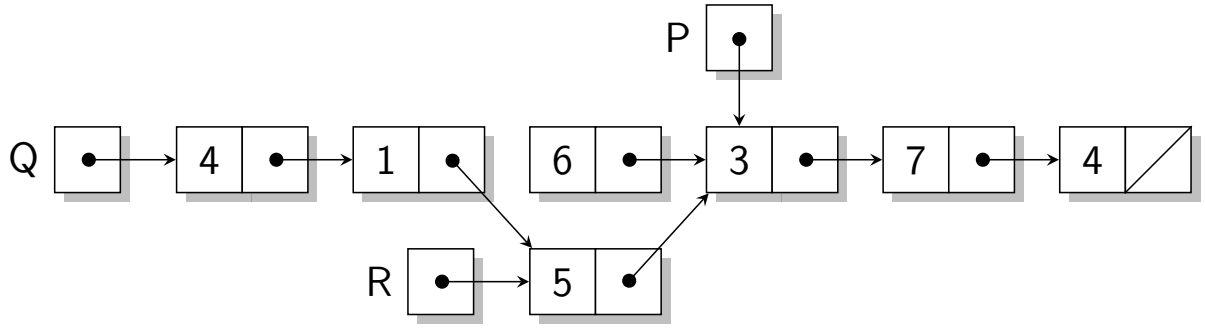
supprimer(Q,Q↑succ) ;



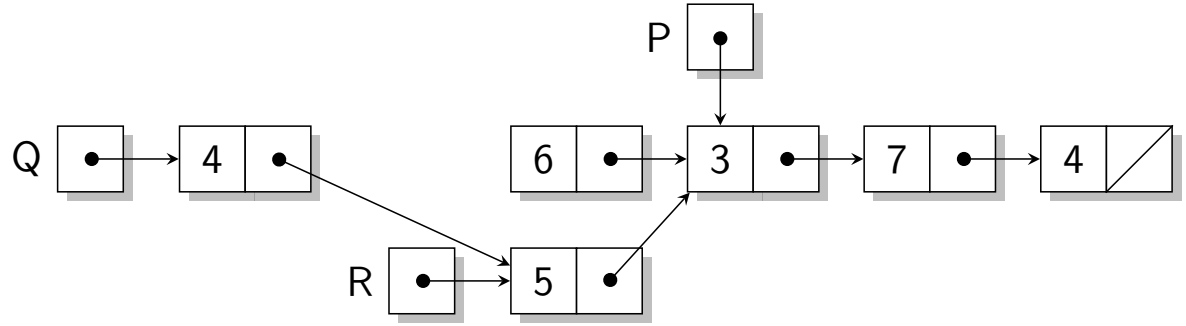
supprimer(Q,Q↑succ) ;



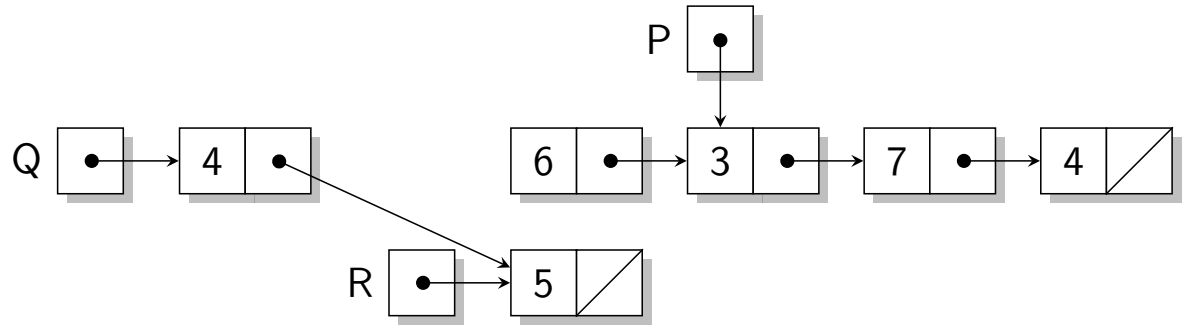
R↑succ ← NULL ;

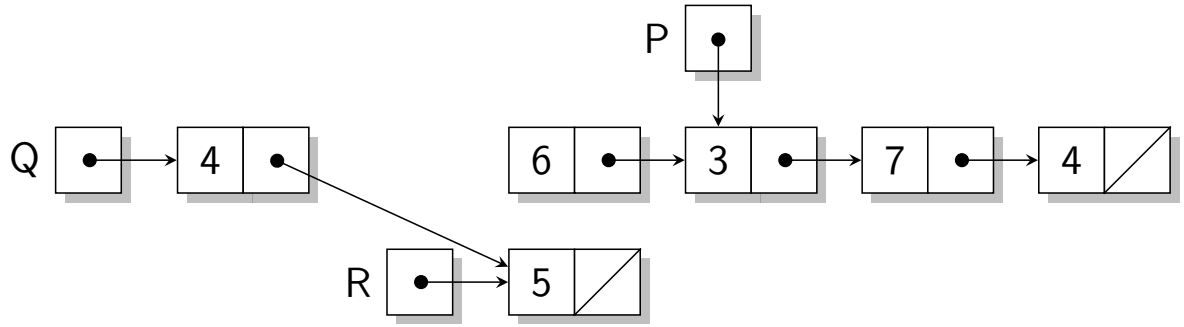


supprimer(Q,Q↑succ) ;

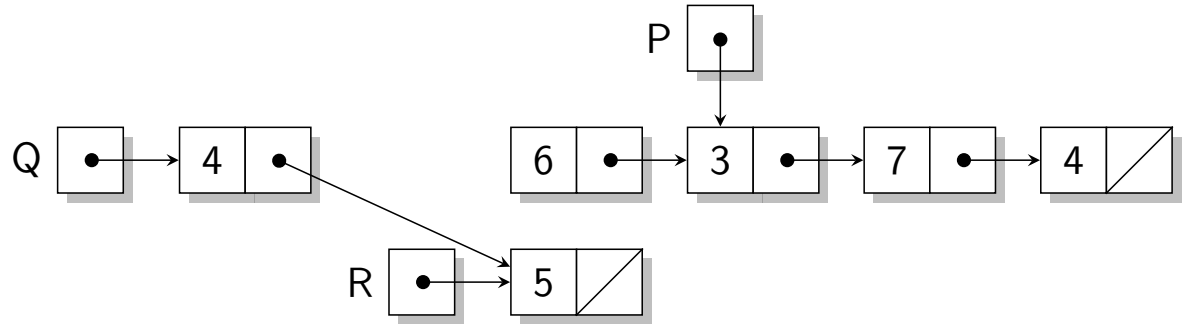


R↑succ ← NULL ;

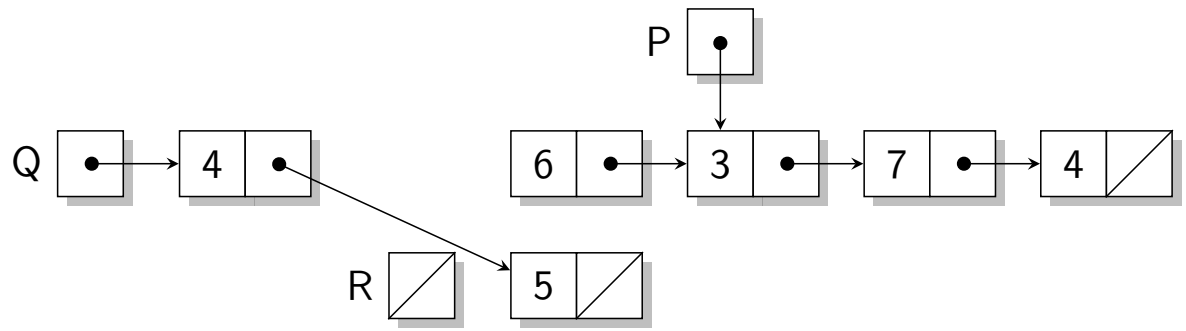




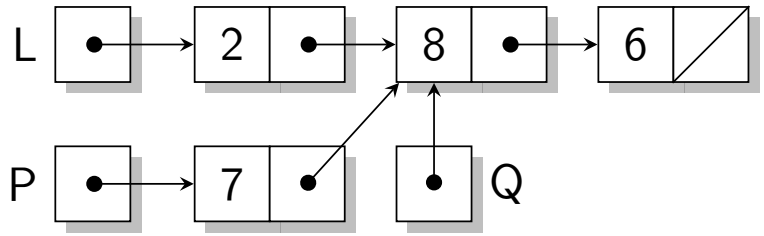
R←NULL ;



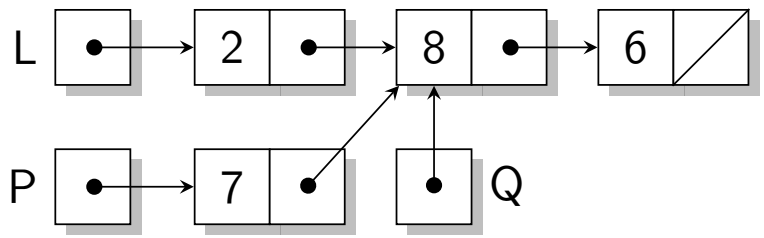
R←NULL ;



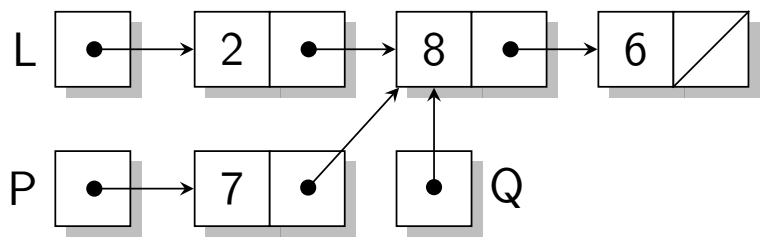
Exercice 2



1. $(Q \uparrow succ) \uparrow info \leftarrow P \uparrow info$

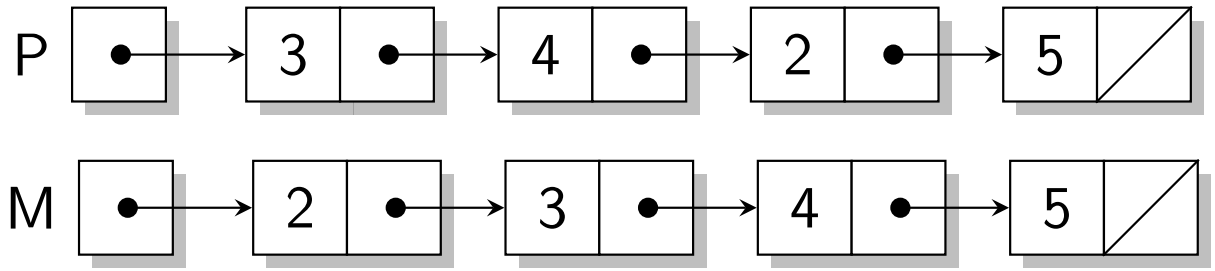


2. $(L \uparrow succ) \leftarrow Q \uparrow succ$



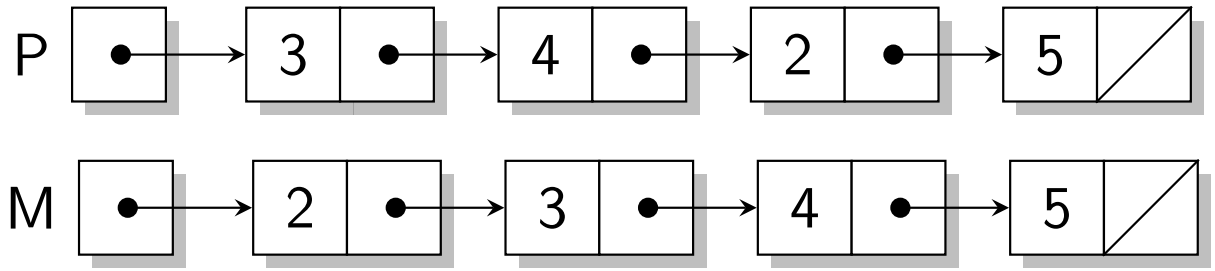
3. insérerAprès$(P, P, 3)$

Exercice 3



Algo : estTrié(d L :liste) : booléen

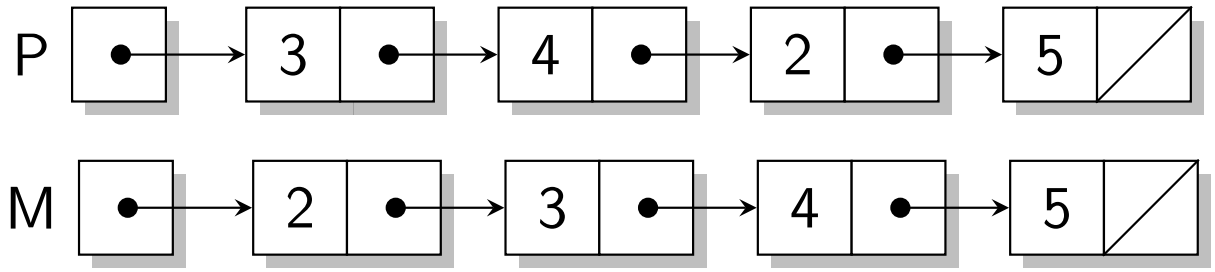
Exercice 3



Algo : estTrié(d L :liste) : booléen

```
si L=NULL ou L↑succ=NULL alors
    renvoyer vrai
sinon si (L↑info) > (L↑succ)↑info alors
        renvoyer faux
    sinon
        renvoyer estTrié(L↑succ)
```

Exercice 3



Algo : estTrié(d L :liste) : booléen

si L=NULL ou L↑succ=NULL **alors**

renvoyer vrai

sinon si (L↑info) > (L↑succ)↑info **alors**

renvoyer faux

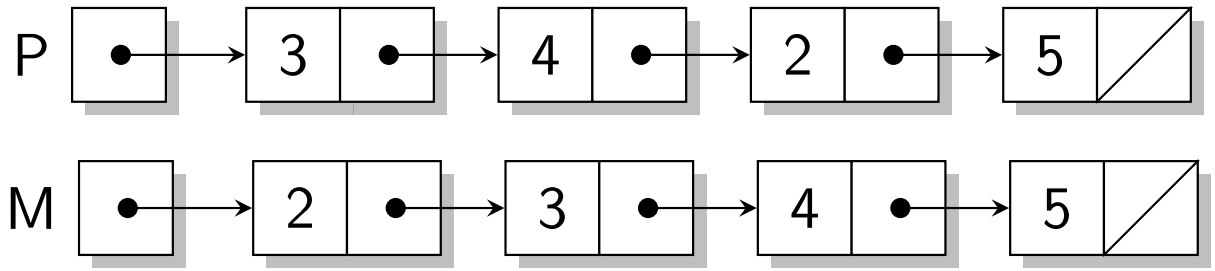
sinon

renvoyer estTrié(L↑succ)

On peut simplifier en

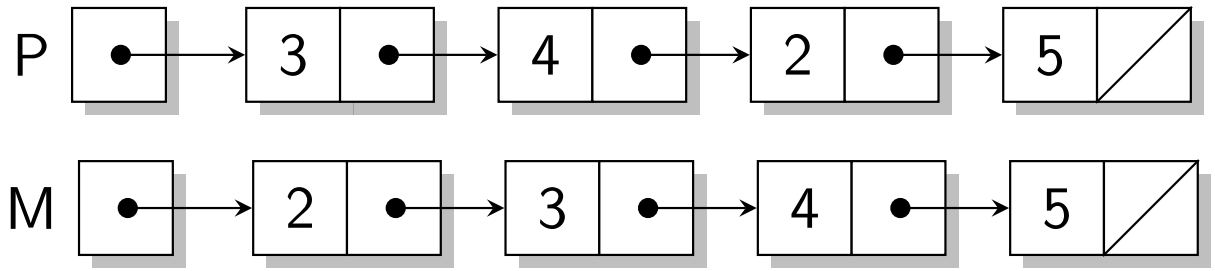
renvoyer L=NULL **ou** L↑succ=NULL **ou (**(L↑info) ≤ (L←succ)↑info **et** estTrié(L↑succ)**)**

Exercice 3



Algo : estTrié(d L :liste) : booléen

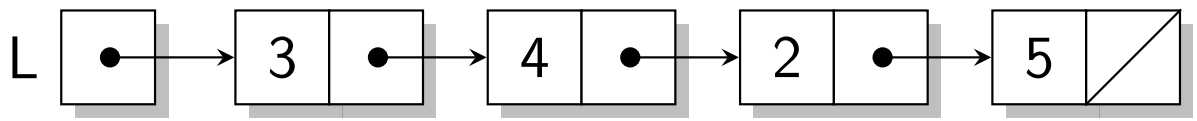
Exercice 3



Algo : estTrié(d L :liste) : booléen

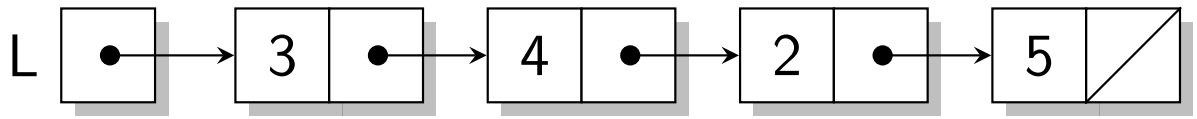
```
variable P : Liste ;
    si L = NULL alors renvoyer vrai
    sinon
        P←L ;
        tant que P↑succ≠NULL et (P↑info) ≤ (P↑succ)↑infoL faire
            P ← P↑succ ;
        fin tq
        renvoyer P↑succ = NULL
```

Exercice 4



Algo : adresseDernier(d L :liste) : Liste

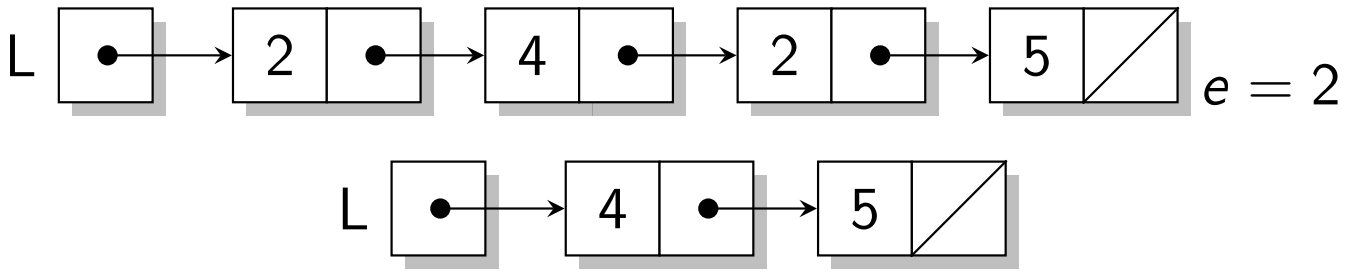
Exercice 4



Algo : adresseDernier(d L :liste) : Liste

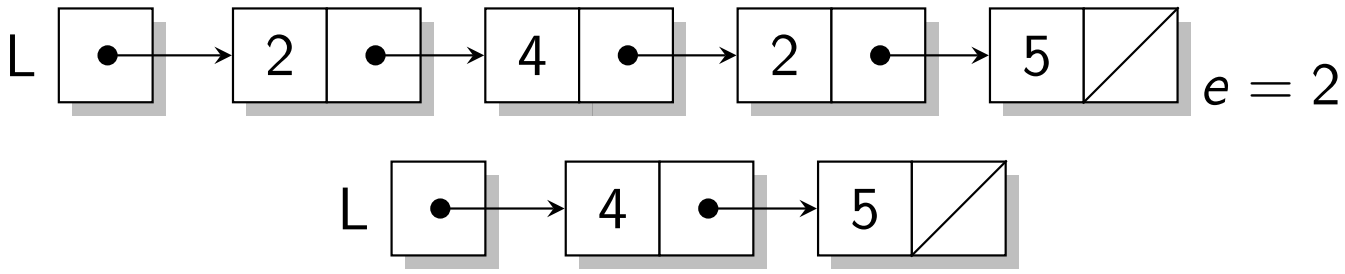
```
si L↑succ=NULL alors
    renvoyer L
sinon
    renvoyer adresseDernier(L↑succ)
```

Exercice 5



Algo : supprimeVal(dr L :liste, d e :entier)

Exercice 5

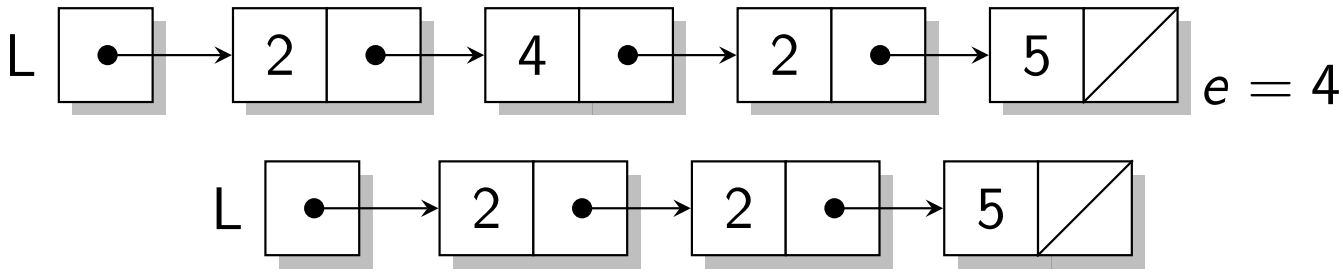


Algo : supprimeVal(dr L :liste, d e :entier)

```
variable M : Liste ;
    M←L
    Tant que M≠ NULL faire
        si M↑info=e alors
            supprimer(L,M)
        sinon
            M←M↑succ
```

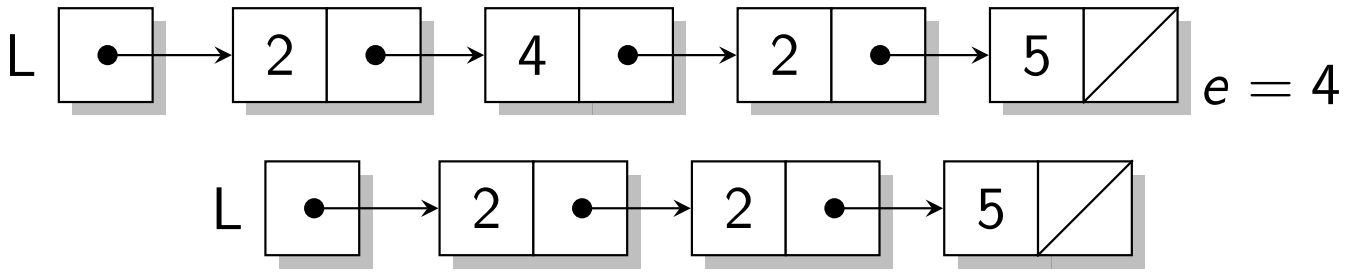
Complexité ?

Exercice 5



Algo : supprimeVal2(dr L :liste, d e :entier)

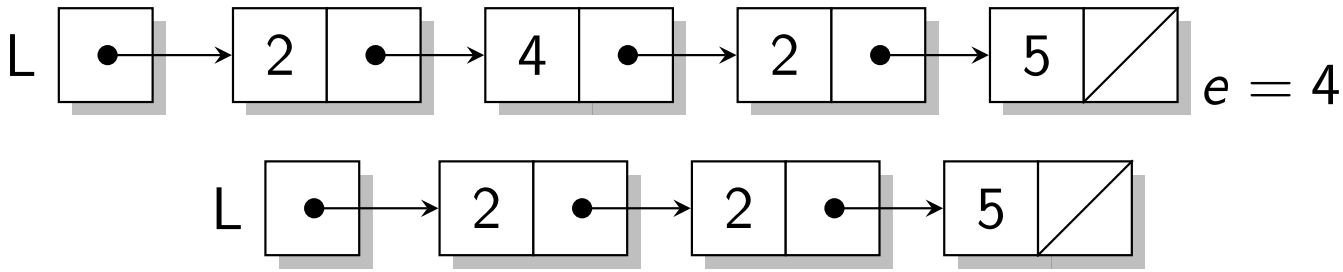
Exercice 5



Algo : supprimeVal2(dr L :liste, d e :entier)

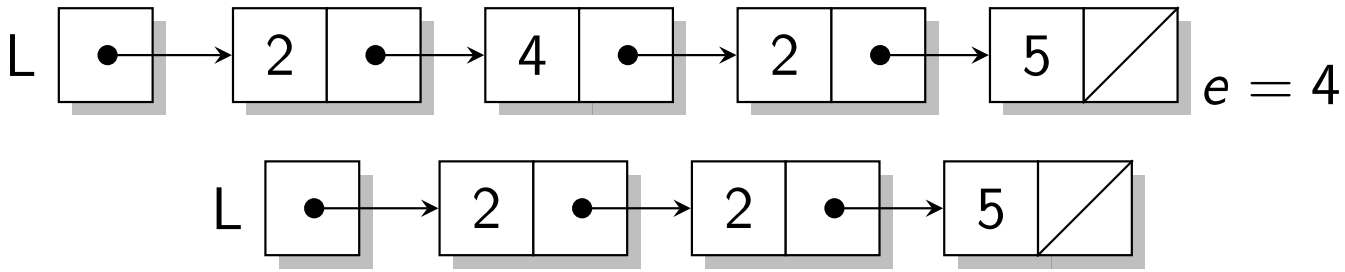
```
    M←L
    Tant que M↑succ ≠ NULL faire
        si (M↑succ)↑info=e alors
            (M↑succ)←(M↑succ)↑succ
        sinon
            M←M↑succ
```

Exercice 5



Algo : supprimeVal2(dr L :liste, d e :entier)

Exercice 5



Algo : supprimeVal2(dr L :liste, d e :entier)

```
    M←L
    Tant que M↑succ ≠ NULL faire
        si (M↑succ)↑info=e alors
            (M↑succ)←(M↑succ)↑succ
        sinon
            M←M↑succ
```