Université de Paris X Nanterre vendredi 12 septembre 2008
Licence MMIA deuxième année
enseignant : L. PIERRE contrôle d'informatique

Qu'écrit le programme suivant quand on l'exécute ? La procédure `affarbreln` dessine des arbres couchés mais, si vous préférez, vous pouvez les dessiner droits.

```
#include<stdio.h>
#include<stdlib.h>
typedef struct noeud *abr;
struct noeud {int cle,sz; abr fg, fd;} vide[1]={{0,1,vide,vide}};
abr equi(abr a);
abr droite(abr a)
{ abr b=a->fg;       a->fg=b->fd;       b->fd=equi(a);
  return b;
}
abr gauche(abr a)
{ abr b=a->fd;       a->fd=b->fg;       b->fg=equi(a);
  return b;
}
abr equi(abr a)
{ if(a->fg->fg->sz>a->fd->sz) return                      equi(droite(a));
  if(a->fg->fd->sz>a->fd->sz) return a->fg=gauche(a->fg), equi(droite(a));
  if(a->fd->fd->sz>a->fg->sz) return                      equi(gauche(a));
  if(a->fd->fg->sz>a->fg->sz) return a->fd=droite(a->fd), equi(gauche(a));
  if(a!=vide) a->sz=a->fg->sz+a->fd->sz;
  return a;
}
abr insere(int x,abr a)
{ if(a==vide) a=malloc(sizeof(*a)), a->cle=x, a->fg=a->fd=vide;
  if(x<a->cle) a->fg=insere(x,a->fg);
  if(x>a->cle) a->fd=insere(x,a->fd);
  return equi(a);
}
abr enleve(int x,abr a)
{ if(a==vide) return a;
  if(x<a->cle) a->fg=enleve(x,a->fg); else
  if(x>a->cle) a->fd=enleve(x,a->fd); else
  if(a->fd==vide) {abr b=a->fg; free(a); return b;} else
  { abr b=a->fd;
    while(b->fg!=vide) b=b->fg;
    a->fd=enleve(a->cle=b->cle,a->fd);
  }
  return equi(a);
}
void aff(abr a) {for(;a!=vide;a=a->fd) aff(a->fg), printf(" %d",a->cle);}
```

```
void affarbre(abr a,int n)
{ for(;a!=vide;a=a->fg)
  affarbre(a->fd,n+=2), printf("%*d\n",n,a->cle);}
void affarbreln(abr a) { printf("\n"), affarbre(a,0); }
typedef struct chainon chainon, *liste;
struct chainon {int val; liste suite;};
liste cree(int val,liste suite)
{ liste a=malloc(sizeof(*a));
  a->val=val;
  a->suite=suite;
  return a;
}
void affl(liste a)
{ for(;a;a=a->suite) printf("%d ",a->val);
  printf("\n");
}
liste a=0,b=0,c=0;
void p1() {liste d=a; a=b; b=d;       }
void p2() {liste d=a; a=c; c=d;       }
void p3() {a->val+=b->val+3;          }
void p4() {b->val+=c->val+4;          }
void p5() {a->suite=cree(10,a->suite);}
void p6() {b       =cree(11,b       );}
void p7() {liste d=a; a=d->suite; d->suite=b; b=d; }
void abc()
{ printf("\n");
  printf("a="); affl(a);
  printf("b="); affl(b);
  printf("c="); affl(c);
}
void (*p[])()={p1,p2,p3,p4,p5,p6,p7};
int main()
{ int t[7]={  ,  ,  ,  ,  ,  ,  }, i;
  abr d=vide;
  for(i=0;i<7;i++) affl(a=cree(t[i],a)), p1(), p2();
  for(i=0;i<7;i++) p[t[i]-1](), printf("p%d",t[i]), abc();
  vide->sz=0;  // Donc dans tout noeud le champ sz sera nul, car 0+0=0
  for(i=0;i!=5;i=(i+7)%12) d=insere(i<7?t[i]:i,d);
  aff(d), affarbreln(d);
  while(d!=vide) d=enleve(d->cle,d), aff(d), affarbreln(d);
  vide->sz=1;  // Réactive le rééquilibrage des arbres
  for(i=0;i<11;i++) d=insere(i<7?t[i]:i,d), aff(d), affarbreln(d);
  return 0;
}
```

Barème sur 22.83 pt

```
listes chaînées : 10.5 pt
  7 premiers affl: 7x(1/2)=3.5 pt
  7 derniers abc : 7x3x(1/3)=7 pt      chaque ligne est juste ou fausse.
ABR : 12.33 pt
  affichage infixe (trié) : 2 pt
  affichage de l'arbre    : 11*(0.33)=3.66 pt
  second affichage        : 10*(0.33)=3.33 pt
  troisième affichage     : 10*(0.33)=3.33 pt
```