Université de Paris ouest Nanterre La Défense mercredi 4 février 2009
Licence MMIA deuxième année
enseignant : L. PIERRE contrôle d'informatique

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

```
#include<stdio.h>
#include<stdlib.h>
typedef int elem;
typedef struct noeud noeud, *abr;
struct noeud { elem clef; abr fg,fd; };
int taille(abr a) { return a ? taille(a->fg)+taille(a->fd) : 1 ; }
void aff(abr a) { for(;a;a=a->fd) aff(a->fg), printf(" %d",a->clef); }
int ta=1, eq=0;   // ta contient la taille de l'arbre, rééquilibrage désactivé
abr nn(abr fg,elem clef,abr fd)
{ abr a=malloc(sizeof(*a)); *a=(noeud){clef,fg,fd}; ++ta; return a; }
void affarbre(abr r)  // affichage sous forme d'arbre couché vers la gauche
{ void af(abr a)      // procédure récursive qui affiche le sous-arbre a de l'arbre r
  { for(;a;a=a->fg)
    { int dd,d=2;    abr p=a->fg;
      a->fg=0, af(a->fd), a->fg=p; // appel récursif à droite sans le fils gauche
      for(p=r;p!=a;p=d?p->fd:p->fg) dd=d,d=!p->fg,printf(dd-!d?"   ":"|  ");
      printf("%c--+  %d\n","\\/-"[d],a->clef);
    }
  }
  af(r);              // unique instruction de affarbre
}
void Aff   (abr a) { aff(a), printf("\n"), affarbre(a); }
abr  rotg  (abr a) { abr b=a->fd; a->fd=b->fg, b->fg=a; return b; }
abr  rotd  (abr a) { abr b=a->fg; a->fg=b->fd, b->fd=a; return b; }
abr equilibre(abr a)     // Rééquilibre l'arbre a en un temps proportionnel à sa taille
{ noeud z={0,0,a}, **b;   int n=taille(a=&z)-1;
  for(b=&a;*b;b=&(*b)->fg) while((*b)->fd) *b=rotg(*b); // a est linéaire à gauche
  for(b=&a;n&(n-1);b=&(*b)->fg,--n) *b=rotd(*b);//tant que n n'est pas une puissance de 2
  while(a!=&z) for(b=&a;*b && (*b)->fg;b=&(*b)->fg) *b=rotd(*b);
  return z.fd;
}
abr bouc(abr a,elem x)
{ int ta=2, s=1;  // s vaut 1 puis 2 puis 3 puis 5 puis 8 puis 12 ... et enfin 0
  abr arrange(abr a)
  { if(x>a->clef) a->fd=arrange(a->fd), ta+=s?taille(a->fg):0; else
    if(x<a->clef) a->fg=arrange(a->fg), ta+=s?taille(a->fd):0;
    s+=(s+1)/2;
    if(s && s>ta) a=equilibre(a), s=0; //On a touvé un bouc émissaire, s nul arrête tout.
    return a;
  }
  return arrange(a);
}
abr insere(elem x,abr a)
{ int s=2;   abr *b=&a;
  while(*b) if((*b)->clef==x) return a;
            else  s+=(s+1)/2, b=x>(*b)->clef? &(*b)->fd : &(*b)->fg;
  *b=nn(0,x,0);
  return s>ta && eq ? bouc(a,x) : a; //branche trop longue : On cherche un bouc émissaire
```

```
}
elem pluspetit(abr a) { while(a->fg) a=a->fg; return a->clef; }
abr enleve(elem x,abr a)
{ abr *b=&a;    // *b est l'arbre duquel enlever x : *b=enleve(x,*b)
  while(*b && (*b)->clef!=x) b=x>(*b)->clef? &(*b)->fd : &(*b)->fg;
  if(*b)        // *b est non nul donc (*b)->cle==x, c'est le noeud à supprimer.
  { abr c=*b;   // Si *b n'a qu'un fils on le remplace par ce fils.
    if(!c->fd) *b=c->fg, free(c), --ta; else
    if(!c->fg) *b=c->fd, free(c), --ta; else // Cette ligne limite la récursivité.
    c->fd=enleve(c->clef=pluspetit(c->fd),c->fd);
  }
  return a;
}
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=c; c=d;       }
void p2() {liste d=a; a=b; b=d;       }
void p3() {a->val+=b->val+1;          }
void p4() {b->val+=c->val+2;          }
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=0;
  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();
  for(i=0;i!=5;i=(i+7)%12) d=insere(i<7?t[i]:i,d);
  Aff(d);
  while(d) d=enleve(d->clef,d), Aff(d);
  eq=1;  // Réactive le rééquilibrage des arbres
  for(i=0;i<11;i++) d=insere(i<7?t[i]:i,d), Aff(d);
  return 0;
}
```

Qu'écrit le programme si on supprime la ligne `if(!c->fg) *b=c->fd, free(c), --ta; else`

Barème sur 21.5 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 : 11 pt
  affichage infixe (trié) : 1 pt
  affichage de l'arbre    : 10*(0.25)=2.5 pt
  second affichage        : 10*(0.25)=2.5 pt
  troisième affichage     : 10*(0.25)=2.5 pt
  second affichage bis    : 10*(0.25)=2.5 pt
```