```c
#include <stdio.h>
#include <stdlib.h>

/*
 * Piles Listes chainées
 * Slide __12__.
 */

//typedef struct personne mon_type;

struct cell {
     int valeur ;
     struct cell * suiv ;
};

typedef struct cell * pile;

void ajout_tete(struct cell ** pL, int x) ;
void sup_tete(struct cell ** pL);

void init_pile(pile * pp) {
     *pp = NULL;
}


void ajout_tete(struct cell ** pL, int x) {
     struct cell * pt;
     pt = malloc(sizeof *pt);
     pt->valeur = x;
     pt->suiv = *pL;
     *pL = pt;
}

int pile_vide2(pile p) { //pas terrible avec les listes contigues car ça
copie tout
     //if (p == NULL) return 1;
     //else return 0;
     return p == NULL; //return 1 si pile est vide
}
int pile_vide(pile *pp) { //avec pointeur
     return *pp == NULL ; //return 1 si pile est vide
}

int pile_pleine(pile p) {
     return 0;
}

void empiler(pile * pp, int v) {
     ajout_tete(pp, v);
}

int dépiler(pile * pp) {
     //pas testé si la pile est vide !!!
     int v = (*pp)->valeur ;
     sup_tete(pp) ;
     return v;
}
```

```c
void dépiler2(pile *pp, int *v) {
     //pas testé si la pile est vide !!!
     *v = (*pp)->valeur ;
     sup_tete(pp) ;
}


//option return
int sommet(pile p) {
     //pas testé si la pile est vide !!!
     return p->valeur ;
}
//option pointeur
void sommet2(pile p, int *v) {
     *v = p->valeur ;
}


void sup_tete(struct cell ** pL){
     struct cell * pt;
     pt = *pL ;
     *pL = (*pL)->suiv ;
     free(pt);
     return ;
}


int main(){
     pile p;
     init_pile(&p);

     empiler(&p, 3);
     empiler(&p, 13);
     empiler(&p, 31);
     empiler(&p, 17);
     empiler(&p, 25);
     empiler(&p, 45);
     empiler(&p, 5);
     empiler(&p, 4);
     empiler(&p, 25);
     empiler(&p, 13);
     empiler(&p, 11);
     empiler(&p, 24);

     int t;
     t = dépiler(&p);
     //dépiler2(&p, &t);

     printf("%d\n", t);

     while(!pile_vide(&p))
          printf("%d\n", dépiler(&p));
     return 0;
}
```