```c
/*
Implantation de l'interface de navigation d'un téléphone portable
à l'aide d'un
automate avec une pile qui mémorise l'état de retour d'une commande

On présente ici un cas simple avec gestion d'appels en cascades.

Notez que

1) le fonctionnement de l'interface est complètement défini dans la
table de transition

2) la fonction qui effectue les transitions est générique au sens où
elle peut s'appliquer à n'importe quelle table de transition qui
repecte les conventions de codage (voir ci-après).

 */

// POUR COMPILER, tapez :  gcc -lncurses -o run main.c

#include <stdio.h>
#include <curses.h>

void initwin(){
  // The recommended ncurses initiation prayer.
  initscr();
  cbreak();
  noecho();
  nonl();
  intrflush(stdscr, FALSE);
  keypad(stdscr, TRUE);
}

void hide_cursor(){
  curs_set(0);
}

void delay(int sec){
  usleep(sec*1000);
}

int get_one_char(){
  int c ;
  nodelay(stdscr, TRUE);  // If we want getch() to return immediatel
y even if no input is available.
  getch();
  scanf("%c",&c);
  return(c);
}

#define Q 6 // les d'états utiles = {1,2,...,Q}
#define A 4 // nombre de symboles de l'alphabet

#define ERREUR (Q+1) // l'état d'erreur
```

```
#define INIT    1    // l'état initial
#define DEFAULT 2    // l'état par défaut en cas de pile vide
#define TOP     0    // indique qu'il faut prendre l'état en sommet
de pile

#define NE Q+1+1 // nombre d'états utiles de {1,...,Q} + l'état 0 (r
eservé) + l'état d'erreur (Q+1)
#define NS A+1 // nombre de symboles de l'alphabet + le symbole rése
rvé ACCEPT
#define TP 9 // taille de la pile = profondeur des menus imbriqués

#define ENTER 10 // La touche ENTER porte le numéro 10
#define ACCEPT 0 // Le status accepteur ou non correspond à la ligne
 0 du tableau Aut

/* CODAGE D'UNE PILE POUR MÉMORISER LES ÉTATS DE RETOUR */

int pile[TP] ;
int s=0 ; // le sommet de la pile

int vider_pile(){
  int i;
  for(i=0;i<TP;i++){ pile[i] = ERREUR ; }
  s=0;
}

int sommet_depiler(){
  int e ;
  if (s>0)
    { e = pile[s] ; s=s-1 ; }
  else
    { e = DEFAULT ; printw("!pile vide!\n") ; }
  return e ;
}

void empiler(int e){
  s=s+1 ;
  pile[s]= e ;
}


/* CODAGE D'UN AUTOMATE COMPLET, DÉTERMNISTE, AVEC MÉMORISATION DES
ÉTATS DE RETOUR

   CONVENTION DE CODAGE

1. La transition par défaut amène dans l'état d'erreur de l'automate
   donc le tableau est au départ initialisé à ERREUR.

2. Codage des transitions qui mémorise l'état de départ dans la pile

   On indique qu'une transition q -l-> q' doit mémoriser son état de
 départ (q)
   dans la pile en notant  (-q', négatif) au lieu de (q', positif) l
'état d'arrivé
```

```
3. Codage des transitions q -l-> (sommet de pile) dont l'état d'arri
vée est l'état
   mémoriser en sommet de pile.

4. On utilise l'état réservé 0 pour indiquer qu'il faut prendre le s
ommet de pile
   comme sommet de retour et dépiler

5. Lorsque la pile est vide, la fonction sommet_depiler retourne tou
jours un état par DEFAULT

6. l'état initial (off) vide la pile.

*/


/* Codage de la correspondance entre
   symboles et numéros des lignes du tableau de l'automate

  Supposons que l'alphabet de l'automate soit {a,s,d}.
  Sachant que le caractère a possède le code ascii 'a'
  et qu'on souhaite indiquer les transitions sur le symbole
  'a' en ligne 1 du tableau Aut ; on indique cette convention
  par l'affectation:
     Symbole['a']=1;
*/

int Symbole[102] ;



/* CODAGE et affichage des modes de l'automate

   Il s'agit de donner des noms (ou mode) au état de l'automate
   Par exemple l'état Q+1 sera nommé "Erreur"
*/

char* Mode[NE] = {"", "1: Off", "2: Accueil", "3", "4", "5", "6", "7
: Erreur"} ;

void affiche_etat(int e){
  printw("\n{%s}\n",Mode[e]) ;
}

void affiche_transition(char c, int tgt){
  printw("%c",c);
  affiche_etat(tgt);
}

/* CODAGE DE l'AUTOMATE */

int Aut[NS][NE] = { // automate de la question b)
  {0,  1,  0,  0,  0,  0,  0,      0 }, // codage du statut des état
s (1=accepteur) : seul l'état initial (Off) est accepteur
```

```
  {0,  2,  3,  4,  5,  6,  1, ERREUR }, // transitions sur 'a' (fonc
tionnalité non spécifiée)
  {0,  6,  1,  2,  3,  4,  5, ERREUR }, // transitions sur 's' (fonc
tionnalité non spécifiée)
  {0, -2, -3, -4, -5, -6, -1, ERREUR }, // transitions sur 'd' (decr
oche comme s mais mémorise l'état précédent)
  {0,  0,  0,  0,  0,  0,  0, ERREUR }  // transitions sur 'r' (racc
roche)
};

// Fonctionnement de l'automate

int transition(int ec, char c, int Aut[NS][NE]){
  int target = Aut[ Symbole[c] ][ec];
  if (target == TOP) { target = sommet_depiler() ; }
  else if
    (target<0) { empiler(ec) ; target = -target ;}

  if (target == INIT) { vider_pile() ; }

  affiche_transition(c,target);
  return target ;
}

int accepteur(int e, int Aut[NS][NE]){
  return Aut[0][e] ;
}

// Exemple d'utilisation : reconnaissance d'un mot entré au clavier

int main(){

 initwin();

 int i;
 for(i=0;i<102;i=i+1){ Symbole[i]=0 ; }
 Symbole['a']=1;
 Symbole['s']=2;
 Symbole['d']=3;
 Symbole['r']=4;

 char c;
 int e;

 vider_pile();

 e = 1; // état initial
 affiche_etat(e);
 c = get_one_char();

 while(c!='q'){
   e = transition(e,c,Aut);
   c = get_one_char();
  }
```

```
 if ( accepteur(e,Aut) )
  printw(":accept\n\n");
 else
  printw(":reject\n\n");

 printw("appuyer sur une touche pour quitter le programme"); getch()
; scanf("%c",&c);

 endwin();

return 0 ;
}
```