Devoir Surveillé - FI

- durée : 1h -
- Tous documents autorisés – Calculatrice interdite –
Composer directement sur ces feuilles

Partie 1 : Tableaux, fonctions, pointeurs (12 points)

a) On souhaite simuler un robot se déplaçant dans un environnement à 2 dimensions sous forme de grille, au moyen du tableau déclaré comme ci-dessous (en variable globale) :

```
#define EST       1
#define NORD      2
#define OUEST     3
#define SUD       4
#define H         7
#define L         7
int ENV[H][L] = { 1,1,1,1,1,1,1,
                  1,0,0,0,1,0,1,
                  1,0,0,0,0,0,1,
                  1,0,0,0,0,0,1,
                  1,1,0,0,0,0,1,
                  1,1,0,0,0,0,1,
                  1,1,1,1,1,1,1};
```

Les 0 correspondent à des cases vides et les 1 à des cases occupées par un obstacle (ou faisant partie d'un mur).

On souhaite visualiser dans une console texte cet environnement ainsi que le robot, en utilisant les caractères suivants :

- vide : ' ' (espace)
- obstacle : '*'
- robot : 'v', '<', '>', ou '^', selon sa direction

Ecrire une fonction réalisant cet affichage, le prototype de cette fonction étant le suivant :

```
//entrees : coordonnees (x_r, y_r) et orientation (o_r) du robot
void affichage_environ(int x_r, int y_r, int o_r);
```

(3 points)

```
void affichage_environ(int x_r, int y_r, int o_r)
{
   int i, j;

   for(j=0 ; j<L_ENV ; j++)
   {
      printf("  ");
      for(i=0 ; i<L_ENV ; i++)
      {
         if(x_r==i && y_r==j)
         {
            if(o_r==ORI_DROITE)          printf(">");
            else if(o_r==ORI_BAS)        printf("v");
            else if(o_r==ORI_GAUCHE)     printf("<");
            else if(o_r==ORI_HAUT)       printf("%c", '^');
         }
         else
         {
          if(ENV[j][i]==0)               printf(" ");
          else if(ENV[j][i]==1)          printf("*");
         }
      }
      printf("\n");
   }
   printf("\n");
}
```

b) Ecrire une fonction *robot_avancer* correspondant au déplacement du robot d'une case dans l'environnement (et donc déterminant ses nouvelles coordonnées), dont le prototype est le suivant :

```
//entree : orientation (o_r) du robot (EST, NORD, OUEST ou SUD)
//sorties : coordonnees (x_r, y_r) du robot
int   robot_avancer(int o_r, int *x_r, int *y_r);
```

Cette fonction devra renvoyer 1 (par retour de la fonction) quand le robot a effectivement avancé (pas d'obstacle devant lui), 0 sinon.

(3 points)

```
int
robot_avancer(int o_r, int *x_r, int *y_r)
{
    if(o_r==EST && ENV[*x_r+1][*y_r]==0)
    {
        (*x_r)++;
        return 1;
    }
    else if(o_r==NORD && ENV[*x_r][*y_r-1]==0)
    {
        (*y_r)--;
        return 1;
    }
    else if(o_r==OUEST && ENV[*x_r-1][*y_r]==0)
    {
        (*x_r)--;
        return 1;
    }
    else if(o_r==SUD && ENV[*x_r+1][*y_r+1]==0)
    {
        (*y_r)++;
        return 1;
    }
    return 0 ;
}
```

c) Ecrire un programme principal appelant cette fonction avec les paramètres suivants :
- orientation du robot : OUEST
- coordonnées du robot : (5, 1)

(et donc utilisant 3 variables) et affichant le résultat de son exécution (à savoir la valeur retournée et les nouvelles coordonnées du robot).

- (2 points)

```
void main()
{
   int o_r=OUEST, x_r=5, y_r=1;
   int retour;

   retour=robot_avancer(o_r, &x_r, &y_r);
   printf("retour=%d, x_r=%d, y_r=%d\n", retour, x_r, y_r) ;
}
```

3. Donner le résultat d'exécution du programme suivant :

(4 points)

```
int main()
{
   char tab[] = {'v', 'r', 'a', 'b', 'o'};
   char *p_tab_1, *p_tab_2, memo;
   int i;

   printf("%c\n", tab[1]);
   p_tab_1=tab+2;
   printf("%c\n", *p_tab_1);
   p_tab_1=p_tab_1-2;
   printf("%c\n", *p_tab_1);
   p_tab_2=tab+3;
   memo=*(tab+3);
   *p_tab_2=*p_tab_1;
   *p_tab_1=memo;
   for(i=0 ; i<5 ; i++)
     printf("%c", tab[i]);
   printf("\n");
   memo++;
   printf("memo=%c\n", memo);
}
```

r	1 point
a	1 point
v	1 point
bravo	0,5 point
memo=c	0,5 point

Partie 2 : Opérations logiques bit-à-bit (8 points)

2.1 Questionnaire à choix multiples (4 points)

Cocher la case située à côté de la réponse choisie
1 point par bonne réponse
-1/2 point par mauvaise réponse

a) Qu'affiche l'extrait de programme suivant ?

```
unsigned char a=0xaa, b=160, c;
c = a & b;
printf("%x\n", c);
```

☐ aa ☐ 160 ■ a0 ☐ 170

b) Qu'affiche l'extrait de programme suivant ?

```
short int a=0x15, b=14, c;
c = a ^ b;
printf("%d\n", c);
```

☐ 4 ☐ -31 ☐ 31 ■ 27

c) Qu'affiche l'extrait de programme suivant ?

```
char a=0x88;
printf("%d\n", a);
```

☐ 88 ☐ 136 ■ -120 ☐ 0,2

d) Qu'affiche l'extrait de programme suivant ?

```
char P1IN=0xaa, P1OUT=255;
if((P1IN & 0x80) == 0x80)
      P1OUT &= ~(0xf0);
printf("%d\n", P1OUT);
```

☐ 255 ☐ f ☐ -8 ■ 15

2.2 Exercices (4 points)

a) Donner le résultat d'exécution du programme suivant, et indiquer l'opération réalisée par la fonction

(2 points)

```
void ma_fonction(int *x, int ib)
{
   unsigned char a;

   a = 1 << ib;
   *x = *x | a;
}
int main()
{
  int a=1;

   ma_fonction(&a, 6) ;
   printf("a=%d\n", a) ;
}
```

Résultat d'exécution :
a=65

Cette fonction force à 1 le bit d'indice ib dans le nombre x

b) On considère un microcontrôleur doté d'un port d'entrée/sortie à 8 bits, programmables par l'intermédiaire de 2 variables (de type int) :

- P1IN : entrées du port
- P1OUT : sorties du port

Un Interrupteur est connecté au bit (d'indice) 0 du port ; une LED est connectée au bit (d'indice) 2 du port.

L'interrupteur provoque une entrée à 1 quand il est activé ; la LED est allumée quand un niveau 1 se trouve sur le bit 7 du port.

On souhaite que la LED s'allume quand l'interrupteur est activé.

Compléter le programme suivant pour obtenir ce fonctionnement.

(2 points)

```
#define BIT0      0x01
#define BIT1      0x02
#define BIT2      0x04

int main()
{
  int P1IN, P1OUT;
  ...
  while(1)
  {
    ...
    if((P1IN & BIT0) == BIT0)    //test interrupteur
    {
        P1OUT |= BIT7;           //allumage LED
    }
  }
}
```