Examen intra
INF3135

Date : Mardi, 13 juin 2006.

Nom: _____
Code permanent: _____

	/	18
	/	18
	/	18
	/	15
	/	6
	/	25
	/	100

Directives pédagogiques :

- Tous les documents sont autorisés.
- Dans toutes les questions de programmation, vous pouvez faire appel sans réserve aux fonctions de la bibliothèque standard, qui incluent entre autres :
  - `stdio.h : sscanf, sprintf, scanf, sprintf, ...`
  - `string.h : strcat, strcpy, strcmp, strlen, ...`
  - `stdlib.h : malloc, free, ...`

Numéro 1. (18 pts) Soit les déclarations suivantes :

```
typedef enum { FAUX = 0, VRAI } Booleen;

typedef enum {
  EXPRESSION_ET,
  EXPRESSION_OU,
  EXPRESSION_NON,
  EXPRESSION_CONSTANTE
} TypeExpression;

typedef struct _expression *Expression;

struct _et {
  Expression gauche;
  Expression droite;
};

struct _ou {
  Expression gauche;
  Expression droite;
};

struct _non {
  Expression droite;
};

struct _constante {
  Booleen v;
};

struct _expression {
  TypeExpression type;
  union {
    struct _et et;
    struct _ou ou;
    struct _non non;
    struct _constante constante;
  } expression;
  Booleen valeur;
  Booleen estEvalue;
};

Expression creerEt( Expression gauche, Expression droite  );
void       creerOu( Expression gauche, Expression droite, Expression *resultat );
```

a) (6 pts) Écrivez le code pour le constructeur `creerEt` déclaré plus haut. Les valeurs `gauche` et `droite` seront assignées à leurs champs respectifs. Les champs `valeur` et `estEvalue` seront initialisés à `FAUX`. N'oubliez pas d'allouer de l'espace mémoire pour la structure.

b) (6 pts) Écrivez le code pour le constructeur `creerOu` déclaré plus haut. Ce constructeur diffère de `creerEt` par ses arguments : la valeur de retour est placée dans une variable transmise *par référence* plutôt que d'être retournée comme résultat de la fonction.

c) (6 pts) Écrivez le code qui appelle le constructeur `creerOu` et place le résultat dans la variable `exp` déclarée ci-dessous. Utilisez la valeur `NULL` pour les arguments `gauche` et `droite` de la routine.

```
int main () {
  Expression exp;

  // Placez votre appel ici :









  // ------------------------

  return 0;
}
```

Numéro 2. (18 pts) Écrivez le code pour la procédure ayant l'en-tête suivant :

```
void insererChaine( char *destination, const char *source, int position );
```

Cette procédure insère la chaîne `source` dans la chaîne `destination` à partir du caractère à la case indiquée par `position`. Par exemple, soit les arguments suivants :

destination : `insererne`

source : `Chai`

position : `8`

Après l'appel, `destination` contiendra : `insererChaine`

Note : c'est la responsabilité *de l'appelant* (donc pas celle de la routine) de s'assurer *i*) que `destination` a assez de place et *ii*) que `position` dénote une position valide à l'intérieur de la chaîne `destination`.

Numéro 3. (18 pts) Soit la déclaration suivante :

```
typedef struct {
  int clef;
  void *element;
} Association;
```

Écrivez le code 'C' qui réalise la fonction de tri suivante :

```
void trier( Association *tableau, int nombreElements ) {
  POUR i allant de 1 à nombreElements - 1 FAIRE
    j ⇐ i - 1
    TANTQUE ( j ≥ 0 ) et ( tableau[j].clef > tableau[j+1].clef ) FAIRE
      tableau[j] ⟺ tableau[j+1]
      j ⇐ j - 1
    FIN
  FIN
}
```

Numéro 4. (15 pts) Que va afficher le programme suivant :

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

int estFaux( char c ) {
  return c == '.';
}

int estVrai( char c ) {
  return c == 'o';
}

int main() {
  char *base = "..........";
  int i = 0, j = 0;
  char *ch1 = (char *) malloc( sizeof(char) * 10 );
  char *ch2 = (char *) malloc( sizeof(char) * 10 );

  strcpy( ch1, base );
  strcpy( ch2, base );

  ch1[4] = 'o';
  ch1[5] = 'o';

  printf( "%s\n", ch1 );

  for( i = 0; i < 5; i ++ ) {
    for( j = 0; j < 10; j++ ) {
      char avant = j == 0 ? '.' : ch1[j-1];
      char courant = ch1[j];
      char apres = j == 9 ? '.' : ch1[j+1];

      if( ( estFaux( avant ) && estFaux( courant ) && estFaux( apres ) )
          ||
          ( estVrai( avant ) && estVrai( courant ) && estVrai( apres ) ) ) {
        ch2[j] = '.';
      } else {
        ch2[j] = 'o';
      }
    }
    printf( "%s\n", ch2 );
    for( j = 0; j < 10; j ++ ) ch1[j] = ch2[j];
  }

  return 0;
}
```

Numéro 5. (6 pts) Que va afficher le programme suivant :

```
#include <stdio.h>
int p;
int q = 100;
int r = 200;

int f( int x ) {
  p += x;
  r += q;
  q += r;
  printf( "%i, %i, %i\n", p, q, r );
  return 2 * p;
}

int h( int x ) {
  int f(int x) {
    p++;
    return x + p;
  }
  int p = f(x);
  return p - 1;
}

int main() {
  int q = p = 5;
  int r = 0;
  int p = 12;

  printf( "%i, %i, %i\n", p, q, r );
  {
    int r = p + 5;
    p++;
    q += 2;
    printf( "%i, %i, %i\n", p, q, r );
  }
  q = f( p );
  printf( "%i, %i, %i\n", p, q, r );

  r = h( p );
  printf( "%i, %i, %i\n", p, q, r );

  f( 0 );

  return 0;
}
```

Remarque pour la correction de ce numéro : Ce programme va afficher six (6) lignes de nombres. La correction se fera de la première ligne vers la dernière. Si une ligne contient une erreur alors les lignes suivantes ne seront pas corrigées. (Par exemple, si la quatrième ligne contient une erreur, alors une note de 3/6 sera attribuée.)

Numéro 6. (25 pts) Un système d'exploitation doit gérer plusieurs fenêtres. Une des routines du système permet de déterminer sur quelle fenêtre se trouve le curseur contrôlé par la souris. Soit les déclarations suivantes :

```
typedef struct {
  int x;
  int y;
} Position;

typedef struct {
  int identificateur;
  Position coin_superieur_gauche;
  Position coin_inferieur_droit;
  int priorite;
} *Descripteur_Fenetre;

Booleen estDansRectangle( Position p, Position coin_sup_gauche, Position coin_inf_droit );
/* Cette fonction retourne VRAI (1) si le point p est à l'intérieur du rectangle
   dénoté par les points coin_sup_gauche et coin_inf_droit.
   Autrement, elle retourne FAUX (0) . */
```

Notre système maintient un tableau de `Descripteur_Fenetre`. Ce tableau est d'une taille fixe de 200 éléments :

```
#define NB_FENETRES 200
Descripteur_Fenetre lesFenetres[NB_FENETRES];
```

Si une case du tableau n'est pas utilisée alors elle contient la valeur `NULL`, sinon elle contient un descripteur de fenêtre valide.

Vous devez écrire une routine qui retourne le champ `identificateur` du descripteur de fenêtre qui représentre la fenêtre où se trouve le curseur de la souris. Le curseur est à l'intérieur d'une fenêtre si les valeurs `x` et `y` de sa `Position` sont comprises inclusivement (donc possiblement égales) dans le rectangle décrit par les `Position`s indiquées dans le descripteur de fenêtre. Si le curseur est à l'intérieur de plusieurs fenêtres, alors se sera la fenêtre qui a la plus grande `priorite` qui sera retournée comme résultat de la fonction.

```
int trouverFenetre( Descripteur_Fenetre *lesFenetres, Position curseur_souris ) {
```