Institut Galilée - Université Paris 13	**Examen du mardi 24 juin 2008**
Licence 1	**Durée : 2 heures. Aucun document autorisé**

INITIATION AUX INTERFACES GRAPHIQUES ET AU WEB

PARTIE 1 : HTML

Exercice 1 : ascendance et descendance. Répondre aux questions suivantes en cochant la ou les case(s) correspondante(s) **et en justifiant vos réponses**.

Un élément "div" peut-il être descendant (direct ou indirect) de :			
head	div	span	*justification*
Un élément "td" peut-il être fils (*ie.*, descendant direct) de :			
table	div	th	*justification*
Un élément "td" qui aurait un unique ascendant "table" peut-il être descendant indirect de :			
table	tr	th	*justification*
Un élément "table" peut-il avoir comme nombre de groupes de lignes :			
0	3	4	*justification*
L'attribut "style" peut-il s'appliquer aux éléments :			
h2	span	style	*justification*

Exercice 2 : tableaux.

Aliment	F	E
Pain	100	15
Fromage	300	45
Total	400	60

Tableau 1

Aliment	**F**	**E**
Pain	100	15
Fromage	300	45
Total	400	60

Tableau 2

Aliment	F	E
Pain	100	15
Fromage	300	45
Total	400	60

Tableau 3

Pour cet exercice, on considère les trois tableaux ci-dessus. On suppose que, par défaut :

- les tableaux s'affichent sans aucun trait de sépération ni de contour ;
- le contenu des cellules "td" est aligné à gauche, en caractères d'épaisseur de trait normale.

Enfin, on rappelle que les attributs "frame" et "rules" de "table" permettent respectivement de définir quels sont les *contours* et quels sont les *séparateurs* de lignes et de colonnes qui seront affichés ; parmi les valeurs que peuvent prendre ces attributs, citons "void" et "box" (pour "frame"), "none", "all" et "groups" (pour "rules").

(1) Produire le tableau 1 en utilisant **exclusivement** les éléments "table", "tr", "td".
(2) Produire le tableau 2 en utilisant **exclusivement** les éléments "table", "tr", "td" et l'attribut "style".
(3) Produire le tableau 3 en utilisant **exclusivement** les éléments "table", "thead", "tbody", "colgroup", "tr", "td", et les attributs "rules", "span".

Exercice 3 : mise en forme. Indiquer, pour chacun des trois codes proposés, la couleur dans laquelle les différents mots sont écrits, et par quelle instruction de mise en forme :

mot	couleur	instruction
code 1		
Titre		
paragraphe		
texte-1		
texte-2		
code 2		
Titre		
texte-1		
paragraphe		
texte-2		
code 3		
Titre-1		
paragraphe-1		
texte-1		
texte-2		
Titre-2		
paragraphe-2		
texte-3		

code 1

entête

```
<style>
   body{color:green;}
   h1{color:blue;}
   div{color:yellow;}
   p{color:red;}
</style>
```

corps

```
<body>
   <h1 style="color:  green;">Titre</h1>
   <div>
      <p>paragraphe</p>
      texte-1
   </div>
   texte-2
</body>
```

code 2

entête

```
<style>
   body{color:green;}
   h1{color:blue;}
   div{color:yellow;}
   p{color:red;}
   body>*{color:purple;}
   h1+div{color:black;}
   div>p{color:white;}
</style>
```

corps

```
<body>
   <h1>Titre</h1>
   <div>
      texte-1
      <p>paragraphe</p>
   </div>
   texte-2
</body>
```

code 3

entête

```
<style>
   .vert{color:green;}
   div{color:red;}
   *{color:blue;}
   div#noir{color:black;}
   div.jaune{color:yellow;}
</style>
```

corps

```
<body>
   <h1>Titre-1</h1>
   <div id="noir">
      <p>paragraphe-1</p>
      texte-1
   </div>
   texte-2
   <h1 class="vert">Titre-2</h1>
   <div>
      <p>paragraphe-2</p>
      texte-3
   </div>
</body>
```

PARTIE 2 : GTK

Exercice 1 : conception d'une fenêtre. On vous demande d'écrire le code GTK permettant de réaliser la fenêtre (principale) donnée en annexe :

(1) Quels sont les objets graphiques GTK impliqués, et comment s'imbriquent-ils (*ex.*, "3 boutons, 2 libellés, le bouton "toto" dans la fenêtre, ...") ?
(2) Écrire le code GTK permettant de déclarer les objets nécessaires (*ie.*, déclarer les variables correspondant aux widgets que vous venez de lister).
(3) Écrire le code GTK permettant de construire les objets (*ie.*, construire pour chaque variable le widget qu'il lui faut, et le lui affecter). Prenez soin de bien renseigner les différents libellés (titre de la fenêtre, libellés, libellés des boutons). Pour les boîtes, on prendra les valeurs TRUE et 0 pour les 2 derniers paramètres de la fonction de construction, TRUE, TRUE et 0 pour les 3 derniers paramètres de la fonction d'ajout.
(4) Écrire le code GTK permettant d'imbriquer les objets entre-eux (*ex.*, ajout d'un widget à la fenêtre).
(5) Écrire le code GTK permettant l'affichage de tous les objets.

Exercice 2 : événementiel.

```
#include <stdio.h>
#include <gtk/gtk.h>

void quitter(GtkWidget*, gpointer);

int main(int argc, char** argv)
{
    GtkWidget* pF = NULL;
    GtkWidget* pB = NULL;
    gtk_init(&argc, &argv);
    pF = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    pB = gtk_button_new_with_mnemonic("_Bouton");
    gtk_container_add(GTK_CONTAINER(pF), pB);
    gtk_widget_show(pB);
    gtk_widget_show(pF);
    gtk_signal_connect(GTK_OBJECT(pF), "destroy", G_CALLBACK(quitter), NULL);
    gtk_main();
    return 0;
}

void quitter(GtkWidget* pW, gpointer pData)
{
    gtk_main_quit();
}
```

On considère le code GTK ci-dessus.

(1) Que se passe-t-il lorsque l'on clique sur le bouton (justifier) ?
(2) Que se passe-t-il lorsque l'on clique sur le coin nord-est de la fenêtre (justifier) ?
(3) Écrire le code GTK permettant que le libellé du bouton devienne "Clicked!" lorsque l'utilisateur clique sur celui-ci (signal "clicked").
(4) Écrire le code GTK permettant que le libellé du bouton devienne "Leave!" lorsque l'utilisateur quitte la zone de celui-ci (signal "leave").
(5) On suppose que le libellé courant du bouton est "Leave!". Que se passe-t-il (*ie.*, quelle(s) instruction(s) sont-elles déclenchées) dans les 4 cas suivants :
   (a) l'utilisateur tappe "ALT+B" au clavier ?
   (b) l'utilisateur tappe "ALT+C" au clavier ?
   (c) l'utilisateur tappe "ALT+L" au clavier ?
   (d) l'utilisateur quitte la zone du bouton ?

Exercice 3.

```
    int main (int argc, char *argv[ ])
    {
 1      GtkWidget* pFenetre = NULL;
 2      GtkWidget* pBoutonValider = NULL;
 3      GtkWidget* pBoutonAnnuler = NULL;
 4      GtkWidget* pBoutonQuitter = NULL;
 5      GtkWidget* pLabelNom = NULL;
 6      GtkWidget* pLabelPrenom = NULL;
 7      GtkWidget* pSaisieNom = NULL;
 8      GtkEntry* pSaisiePrenom = NULL;
 9      gtk_init(NULL, NULL);
10      pFenetre = gtk_window_new(GTK_WINDOW_TOPLEVEL);
11      pBoutonValider = gtk_button_new_with_mnemonic("_Valider");
12      pBoutonAnnuler = gtk_button_new_with_mnemonic("_Annuler");
13      pBoutonQuitter = gtk_button_new_with_mnemonic("_Quitter");
14      pLabelNom = gtk_button_new_with_mnemonic("_Nom");
15      pLabelPrenom = gtk_button_new_with_mnemonic("_Prenom");
16      pSaisieNom = gtk_entry_new();
17      pSaisiePrenom = gtk_entry_new();
18      gtk_label_set_label(GTK_LABEL(pLabelNom), "_Nom");
19      gtk_widget_set_sensitive(pBoutonValider, FALSE);
20      gtk_button_set_relief(GTK_BUTTON(pBoutonValider), GTK_RELIEF_HALF);
21      gtk_button_set_relief(GTK_BUTTON(pLabelNom), GTK_RELIEF_HALF);
22      gtk_entry_set_max_length(pSaisieNom, 100);
23      gtk_entry_set_max_length(pSaisiePrenom, 50);
24      gtk_entry_set_visibility(GTK_ENTRY(pSaisieNom), TRUE);
25      gtk_entry_set_visibility(GTK_ENTRY(pSaisiePrenom), TRUE);
26      gtk_container_add(GTK_CONTAINER(pFenetre), pBoutonValider);
27      gtk_container_add(GTK_CONTAINER(pFenetre), pBoutonAnnuler);
28      gtk_container_add(GTK_CONTAINER(pFenetre), pBoutonQuitter);
29      GtkBin* pBin1 = GTK_BIN(pBoutonQuitter);
30      GtkBox* pBox1 = GTK_BOX(pBoutonQuitter);
31      GtkWidget* pWidget1 = GTK_WIDGET(pBoutonQuitter);
32      GtkBin* pBin2 = pBoutonValider;
33      GtkBox* pBox2 = pBoutonValider;
34      GtkWidget* pWidget2 = pBoutonValider;
35      gtk_box_pack_start(pBox2, pLabelNom, TRUE, 0, 0);
36      gtk_widget_set_sensitive(pWidget2, TRUE);
37      gtk_dialog_run(GTK_DIALOG(pFenetre));
38      return 0;
    }
```

On vous demande de vérifier que le code GTK ci-dessus est *techniquement* correct (*ie.*, sans préjuger des intentions du programmeur). Plus précisément, pour chacune des lignes suivantes, indiquez **en justifiant** si elle produit ou non un avertissement à la compilation, une erreur (ou un avertissement) à l'exécution :

ligne	***compilation***	***exécution***
14		
16		
17		
18		
21		
22		
23		
24		
26		
27		
29		
30		
31		
32		
33		
34		
35		
36		
37		

Annexe

HTML : propriétés.

nom	signification	affectation
font-weight	épaisseur de trait	normal, bold, bolder
text-align	alignement du contenu	left, right, center

GTK : arborescence des objets.

```
GObject
    GtkObject
        GtkWidget
            GtkContainer
                GtkBin
                    GtkWindow
                        GtkDialog
                    GtkButton
                GtkBox
                    GtkHbox
                    GtkVBox
            GtkEntry
            GtkMisc
                GtkLabel
```

GTK : prototypes.

```
GtkWidget* gtk_window_new(GtkWindowType type);
GtkWidget* gtk_label_new(const gchar *str);
GtkWidget* gtk_button_new_with_mnemonic(const gchar *label);
GtkWidget* gtk_entry_new(void);
GtkWidget* gtk_hbox_new(gboolean homogeneous, gint spacing);
GtkWidget* gtk_vbox_new(gboolean homogeneous, gint spacing);
void gtk_widget_set_sensitive(GtkWidget *widget, gboolean sensitive);
void gtk_window_set_title(GtkWindow *window, const gchar *title);
void gtk_label_set_label(GtkLabel *label, const gchar *str);
void gtk_button_set_relief(GtkButton *button, GtkReliefStyle newstyle);
void gtk_entry_set_max_length(GtkEntry *entry, gint max);
void gtk_entry_set_visibility(GtkEntry *entry, gboolean visible);
void gtk_container_add(GtkContainer *container, GtkWidget *widget);
void gtk_box_pack_start(GtkBox *box, GtkWidget *child, gboolean expand, gboolean fill, guint padding);
void gtk_widget_show_all(Gtkwidget *widget);
gint gtk_dialog_run(GtkDialog *dialog);
```

GTK : types, constantes, macros.

GTK_WINDOW_TOPLEVEL est une constante du type énuméré GtkWindowType.
GTK_RELIEF_HALF est une constante du type énuméré GtkReliefStyle.
FALSE est une constanet symbolique de valeur 0, TRUE est la négation de FALSE.
gboolean est un alias de gint, lui-même alias de int ; guint est un alias de unsigned int ; gchar est un alias de char.