LANGAGE C – TP 1
Types – Entrées/sorties standard

Les codes proposés devront impérativement avoir la structure définie dans la fiche de TP1 et des commentaires pertinents seront ajoutés. Une grande importance doit être apportée à la lisibilité des codes.

Exercice 1

Ecrire un programme qui demande à un utilisateur son âge, stocke cet âge dans une variable de type int puis affiche à l'écran l'âge qui a été saisi, en base 10, en base 8 puis en base 16.

Rajeunissez ensuite l'âge de 5 ans et indiquez à l'utilisateur qu'il paraît plutôt ce nouvel âge.

Le rendu à l'écran devra impérativement être le suivant :

```
Bonjour, quel age avez-vous ?  62
Vous avez 62 ans
En octal, cela fait : 76 ans
En hexadecimal, cela fait 3e ans

Vous en paraissez 57...

Appuyez sur une touche pour continuer...
```

Exercice 2

Pour la dénomination des types des entiers de la bibliothèque stdint, le préfixe « u » signifie unsigned, soit « sans signe » ; il s'agit donc d'entiers positifs. L'absence du préfixe « u » dénote un entier relatif.

Pour les différents types d'entiers indiqués dans la fiche de TP, répertoriez le nombre d'octets nécessaires à leur stockage, leurs valeurs minimale et maximale ainsi que le nombre de valeurs pouvant être codées (on ne comptera qu'une seule fois le 0).

Pour cela, générez la sortie suivante (les valeurs des nombres d'octets ont volontairement été escamotées) :

```
Pour les int8_t :
        uint8_t : valeur de 8       stocke sur  octet
        int8_t : valeur de -8       stocke sur  octet

Pour les uint32 :
        uint32_t : valeur de 32       stocke sur  octets
        int32_t : valeur de -32       stocke sur  octets

Pour les uint64 :
        uint64_t : valeur de 64       stocke sur  octets
        int64_t : valeur de -64       stocke sur  octets

Appuyez sur une touche pour continuer...
```

Remarques :
Lorsque plusieurs variables doivent être affichées, on adopte la syntaxe suivante :
printf (‘’ Deux variables a afficher : %d et %d’’, var1, var2) ;
Les variables sont indiquées dans l'ordre où les formats apparaissent dans la chaîne de caractères et sont séparées par une virgule.

L'instruction `sizeof (nomVariable)` renvoie le nombre d'octets nécessaires au stockage de la variable nomVariable.

Exercice 3

On s'intéresse au programme suivant (disponible dans les ressources pédagogiques) :

```
/* TP1 - exercice 3
SP- 09/22 - Depassement
*/

# include <stdio.h>
# include <stdint.h>

/* PROGRAMME PRINCIPAL  */

int main()
{
	/* Déclaration des variables*/

	uint8_t int1 = 255;
	uint8_t int2 = 259;
	uint8_t int3;
	uint8_t int3 = 0;
	uint8_t int4;
	uint8_t int 5 = 2;

	/* Première partie */

	printf("Premier entier %d stocke sur %d octet\n", int1, sizeof (int1));
	printf("Deuxieme entier %d stocke sur %d octet\n", int2, sizeof(int2));
	int4 = int1 + int2;
	printf("Somme de deux entiers :int1 + int2 = %d stocke sur %d octet\n", int4,
sizeof(int4));

	printf("\n\n");
	return 0;
}
```

1) Lors de l'exécution, la génération échoue et les messages (attention : suivant les compilateurs, les messages peuvent être différents) suivants sont fournis :

```
main.c: In function ‘main’:
main.c:18:17: warning: unsigned conversion from ‘int’ to ‘uint8_t’ {aka ‘unsigned char’} changes value from ‘259’ to ‘3’ [-Woverflow]
   18 |  uint8_t int2 = 259;
      |                 ^~~
main.c:21:10: error: two or more data types in declaration specifiers
   21 |  uint8_t int 5 = 2;
      |          ^~~
main.c:21:14: error: expected identifier or ‘(’ before numeric constant
   21 |  uint8_t int 5 = 2;
      |              ^
main.c:26:40: warning: format ‘%d’ expects argument of type ‘int’, but argument 3 has type ‘long unsigned int’ [-Wformat=]
   26 |  printf("Premier entier %d stocke sur %d octet\n", int1, sizeof (int1));
      |                                       ~^                ~~~~~~~~~~~~~~
      |                                        |                |
      |                                        int              long unsigned int
      |                                       %ld
main.c:27:41: warning: format ‘%d’ expects argument of type ‘int’, but argument 3 has type ‘long unsigned int’ [-Wformat=]
   27 |  printf("Deuxieme entier %d stocke sur %d octet\n", int2, sizeof(int2));
      |                                       ~^                ~~~~~~~~~~~~~~
      |                                        |                |
      |                                        int              long unsigned int
      |                                       %ld
main.c:28:54: error: ‘int5’ undeclared (first use in this function); did you mean ‘int4’?
   28 |  printf("Cinquieme entier %d stocke sur %d octet\n", int5, sizeof(int5));
      |                                                      ^~~~
      |                                                      int4
main.c:28:54: note: each undeclared identifier is reported only once for each function it appears in
main.c:30:62: warning: format ‘%d’ expects argument of type ‘int’, but argument 3 has type ‘long unsigned int’ [-Wformat=]
   30 |  printf("Somme de deux entiers :int1 + int2 = %d stocke sur %d octet\n", int4, sizeof(int4));
      |                                                        ~^                 ~~~~~~~~~~~~~~
      |                                                         |                 |
      |                                                         int               long unsigned int
      |                                                        %ld
main.c: In function ‘main’:
main.c:18:17: warning: unsigned conversion from ‘int’ to ‘uint8_t’ {aka ‘unsigned char’} changes value from ‘259’ to ‘3’ [-Woverflow]
   18 |  uint8_t int2 = 259;
      |                 ^~~
main.c:22:10: error: two or more data types in declaration specifiers
   22 |  uint8_t int 5 = 2; //Pour les messages d'erreurs mais inutile ensuite
      |          ^~~
main.c:22:14: error: expected identifier or ‘(’ before numeric constant
   22 |  uint8_t int 5 = 2; //Pour les messages d'erreurs mais inutile ensuite
      |              ^
main.c:28:40: warning: format ‘%d’ expects argument of type ‘int’, but argument 3 has type ‘long unsigned int’ [-Wformat=]
   28 |  printf("Premier entier %d stocke sur %d octet\n", int1, sizeof (int1));
      |                                       ~^                ~~~~~~~~~~~~~~
      |                                        |                |
      |                                        int              long unsigned int
      |                                       %ld
main.c:29:41: warning: format ‘%d’ expects argument of type ‘int’, but argument 3 has type ‘long unsigned int’ [-Wformat=]
   29 |  printf("Deuxieme entier %d stocke sur %d octet\n", int2, sizeof(int2));
      |                                       ~^                ~~~~~~~~~~~~~~
      |                                        |                |
      |                                        int              long unsigned int
      |                                       %ld
main.c:30:42: warning: format ‘%d’ expects argument of type ‘int’, but argument 3 has type ‘long unsigned int’ [-Wformat=]
   30 |  printf("Cinquieme entier %d stocke sur %d octet\n", int5, sizeof(int5));
      |                                       ~^                ~~~~~~~~~~~~~~
      |                                        |                |
      |                                        int              long unsigned int
      |                                       %ld
main.c:32:62: warning: format ‘%d’ expects argument of type ‘int’, but argument 3 has type ‘long unsigned int’ [-Wformat=]
   32 |  printf("Somme de deux entiers :int1 + int2 = %d stocke sur %d octet\n", int4, sizeof(int4));
      |                                                        ~^                 ~~~~~~~~~~~~~~
      |                                                         |                 |
      |                                                         int               long unsigned int
      |                                                        %ld
```

Expliquer chacun des messages d’erreur et avertissement.

2) Corriger le code pour supprimer les erreurs (mais pas les avertissements) et observer le comportement des variables.

3) Indiquer les résultats attendus.
4) Expliquer les résultats obtenus en réalisant les conversions et additions en binaire.
5) On change la valeur de int5 pour l'affecter à -5 et on ajoute une ligne d'affichage, identique à celle écrite pour int1. Prédire le comportement du programme (erreur, warning, valeur affichée ?)
6) Valider votre prévision en générant ces lignes de code.
7) Expliquer le résultat obtenu : pour cela faites une recherche rapide sur internet sur le codage en complément à 2.
8) Conclure.

Exercice 4

On s'intéresse au programme suivant :

```c
/* TP1 - exercice 4
SP- 09/22 - Conversion de types d'entiers
*/

/*-------------------------------------------------------------*/
/* IMPORTATION BIBLIOTHEQUES                                   */
/*-------------------------------------------------------------*/

# include <stdio.h>

/*-------------------------------------------------------------*/
/* PROGRAMME PRINCIPAL                                         */
/*-------------------------------------------------------------*/

int main()
{
	/*--------------------------------------*/
	/* Déclaration des variables*/
	unsigned short int1 = 10, int2 = 32767;
	unsigned int int3 = 100, int4 = 69755;
	float float1 = 15., float2 = 6;
	unsigned short resultatShort;
	unsigned int resultatInt;
	float resultatFloat;

	/*--------------------------------------*/
	/* Instructions*/
	//Affichage préliminaire
	printf("Etat initial des variables\n");
	printf("--------------------------\n");
	printf("Short int1 : %d de taille %d octets\n", int1, sizeof(int1));
	printf("Short int2 : %d de taille %d octets\n", int2, sizeof(int2));
	printf("Int int3 : %d de taille %d octets\n", int3, sizeof(int3));
	printf("Int int4 : %d de taille %d octets\n", int4, sizeof(int4));
	printf("Float float1 : %f de taille %d octets\n", float1, sizeof(float1));
	printf("Float float2 : %f de taille %d octets\n", float2, sizeof(float2));

	printf("\n\n");

	// Opérations diverses et affichages
	printf("Resultats d'operations et de conversions\n");
	printf("----------------------------------------\n");

	resultatShort = int1 / int2;
	printf("Short <- int1 / int2, soit %d / %d = %d\n", int1, int2, resultatShort);
```

```
	resultatFloat = int1 / int2;
	printf("Float <- int1 / int2, soit %d / %d = %f \n", int1, int2, resultatFloat);

	resultatFloat = int1 / float(int2);
	printf("Float <- int1 / float(int2), soit %d / %d = %f\n", int1, int2, resultatFloat);


	printf("\n");
	resultatShort = int3 / float1;
	printf("Short <- int3 / float1, soit %d / %f = %d\n", int3, float1, resultatShort);

	resultatFloat = int3 / float1;
	printf("Float <- int3 / float1, soit %d / %f = %f\n", int3, float1, resultatFloat);

	printf("\n");
	resultatShort = (short)int4;
	printf("Conversion de int4 = %d en short : %d\n", int4, resultatShort);

	printf("\n");
	resultatInt = int4 % int3;
	printf("Int <- int4 modulo int3, soit %d (mod) %d = %d de taille %d octets\n", int4,
int3, resultatInt);

	printf("\n\n");
	return 0;
}
```

La sortie est alors la suivante (le résultat de la dernière ligne peut différé suivant les compilateurs) :

```
Short int2 : 32767 de taille 2 octets
Int int3 : 100 de taille 4 octets
Int int4 : 69755 de taille 4 octets
Float float1 : 15.000000 de taille 4 octets
Float float2 : 6.000000 de taille 4 octets


Resultats d'operations et de conversions
----------------------------------------
Short <- int1 / int2, soit 10 / 32767 = 0
Float <- int1 / int2, soit 10 / 32767 = 0.000000
Float <- int1 / float(int2), soit 10 / 32767 = 0.000305

Short <- int3 / float1, soit 100 / 15.000000 = 6
Float <- int3 / float1, soit 100 / 15.000000 = 6.666667

Conversion de int4 = 69755 en short : 4219

Int <- int4 modulo int3, soit 69755 (mod) 100 = 55 de taille 10 octets
```

Les codages en binaire de ces différents nombres sont :

Base 10	Binaire
10	00000000 00001010
32767	11111111 11111111
100	00000000 00000000 00000000 01100100
69755	00000000 00000001 00010000 01111011

1) Division
    a. Comparer les résultats des deux premières divisions. En déduire l'ordre dans lequel se font les opérations / et les conversions de type.
    b. Quel est l'intérêt de la conversion de int2 en float dans la troisième division ?
    c. Expliquez les résultats obtenus pour les deux dernières divisions.

2) Conversion

Justifier le résultat obtenu pour la conversion de int vers short de la variable int4.

3) Modulo
    a. Indiquer ce que réalise l'opérateur %.
    b. Quel est le type retourné par cette opération mathématique ?
    c. Expliquez le nombre d'octets occupés en mémoire annoncé pour le résultat de la dernière opération.
    d. On ajoute au programme les 2 lignes suivantes :

```
resultatInt = float2 % float1;
printf("Int <- float2 modulo float1, soit %d (mod) %d = %d de taille %d octets\n", float2,
float1, resultatInt);
```

Ces lignes provoquent le message d'erreur suivant :

```
main.c:69:23: error: invalid operands to binary % (have 'float' and 'float')
   69 |  resultatInt = float2 % float1;
```

Commenter ce message.