Exercice 1 :

1) Le modèle ILP64 :

   int = 4 bits, longint = 8 bits, * = 8 bits

2) sizeof(longint) et sizeof(T*) changent,
   a. La taille de toutes structures qui utilisent ces types changent.
   b. Si un longint utilise une valeur > max (int), le calcul est incorrecte.
   c. Un pointeur 32 bits ne permet d'accéder qu'a 4GB de mémoire, donc ci le programme alloue plus de 2GB de mémoire = échec.
3) Avoir des entiers de taille fixe permet de construire des structures ayant une taille fixe sur tous les systèmes.

Exercice 2 :

1) Lors d'une promotion numérique, la conversion d'un type en un autre ce fait sans perte de précision.
2) Une promotion numérique ce fait toujours sans perte de précision,
   a. exemple : entier 16 bits -> entier 32 bits.
3) x modulo 2^p où p est le nombre de bits du type vers lequel on convertit.

4) Valeur :
   a. r1 = (100 * 3) mod 256 = 300 mod 256 = 44
   b. r2 = 44/4 = 11
   c. r3 = (100 * ¾) mod 256 = 300/4 mod 256 = 75
   d. Règle : tout calcul entier sur de types de taille < int se fait en int puis est reconvertit.
5) Oui, si le nombre de bits significatif de l'entier dépasse le nombre de bits de la mantisse du flottant.
6) Oui, il suffit que l'exposant du flottant soir > 10^9 ou que ce soit un nombre fractionnaire.

7) Conversion :

   a. flottant -> entier : 3.2 => 3 -3.2 => -3

   b. floorf 3.2 => 3 -3.2 => -4 conversion vers le bas

   c. ceilf 3.2 => 4 -3.2 => -3 conversion vers le haut

Exercice 3 :

(lvalue / rvalue ici sens à partir du c_{11}++)

1)
   a. lvalue = une expression identifiable (= on peut récupérer l'adresse)
   b. rvalue = une expression déplaçable
   c. Par défaut Une lvalue n'est pas déplaçable
      Une rvalue n'est pas identifiable
2)
   a. Sans modificateur = variable lvalue
   b. & = référence à une lvalue
   c. && = référence à une rvalue
   d. * = pointeur (-> type diffèrent)
3) Elle est nommée donc c'est une lvalue
4) Code 1 :

int a = 5;	lvalue I/nD	rvalue nI/D	oui	rvalue copie dans lvalue
int &b = a;	lvalue I/nD	lvalue I/nD	oui	réf lvalue depuis lvalue
int c = b;	lvalue I/nD	lvalue I/nD	oui	lvalue copié dans réf lvalue
int d = (a+4)/2;	lvalue I/nD	rvalue nI/D	oui	rvalue copié dans lvalue
int &e = 5;	lvalue I/nD	rvalue nI/D	non	rvalue -> réf lvalue
int &f = a/2;	lvalue I/nD	rvalue nI/D	non	idem
int &&g = a;	lvalue I/nD	lvalue I/nD	non	lvalue -> réf rvalue
int &&h = b;	lvalue I/nD	lvalue I/nD	non	idem
int &&i = a/4;	lvalue I/nD	rvalue nI/D	oui	réf lvalue depuis rvalue
int &&j = 8;	lvalue I/nD	rvalue nI/D	oui	idem
	gauche	droite	val ?	

I = identifiable D = Déplaçable nX = non X

5) Code 2 :

int A = fun1()	idem	rvalue nI/D	oui	rvalue copié dans lvalue
int &B = fun2(a)	idem	lvalue I/nD	oui	réf lvalue depuis lvalue
int &&C = fun3(5)	idem	xvalue I/D	oui	réf rvalue depuis xvalue
int D = fun2(a)	idem	lvalue I/nD	oui	réf lvalue copié dans lvalue
int E = fun3(7)	idem	xvalue I/D	oui	réf rvalue copié dans lvalue
int &F = fun3(a)	idem	xvalue I/D	non	réf rvalue -> réf lvalue
int &G = fun1(a)	idem	rvalue nI/D	non	rvalue -> réf lvalue
int &&H = fun2(a)	idem	lvalue I/nD	non	réf lvalue -> réf rvalue
int &&I = fun1()	idem	rvalue nI/D	oui	rvalue vers réf rvalue

Int fun1 (), int &fun2 (), int &&fun3 ()

6)
   a. const int = entier constant
   b. const int& = réf vers un entier constant

c. const int&& = réf vers un entier temporaire constant

7)

const int &e = 5;	lvalue I/nD	rvalue nI/D	oui
const int &f = a/2;	lvalue I/nD	rvalue nI/D	oui

Int fun (int &a) {}

➔ fun (b) valide
➔ fun (4) invalide

Int fun (const int &a) {}

➔ fun (b) valide
➔ fun (4) valide

Exercice 6 :

1) Règles de surcharge :
   a. Au sein d'une même catégorie, une surcharge est conflictuelle.
   b. Un paramètre de type value ne peut pas être surchargé par un paramètre de type lvalue ou rvalue.
   c. Un paramètre de type lvalue peut être surchargé par un paramètre de type rvalue.

2)
   a. non, règle 1 (conflit, surcharge même coté)
   b. non, règle 2 (pas de surcharge de value vers lvalue)
   c. non, règle 2 (pas de surcharge de value vers rvalue)
   d. non, règle 2 (pas de surcharge de value vers lvalue)
   e. non, règle1
   f. oui
      i. fun (int&) capte les lvalue
      ii. fun (int&&) capte les rvalue
   g. oui
      i. fun (int &) capte les lvalue
      ii. fun (const int&&) capte les rvalue + lvalue const
   h. oui
      i. capte les lvalue + lvalue const
      ii. capte rvalue
   i. oui
      i. capte les lvalue + lvalue const
      ii. capte rvalue + rvalue const
   j. non, règle 1