3TC31 (ex. INF107)

Examen intermédiaire (deuxième partie)

2025–2026

Nom : .....

Prénom : .....

Instructions
- Durée : 30 minutes
- Document autorisé : 1 feuille A4 recto-verso uniquement
- Dispositifs électroniques (calculatrice, ordinateur...) interdits
- Vous pouvez répondre en français ou en anglais
- Ce sujet contient quelques questions de cours et deux exercices indépendants
- **Répondez directement sur le sujet**

Exercice 1 : Questions de cours (4 points / 4 minutes)

Question 1 (2 points)

Quel est le lien entre la durée de vie (*storage duration*) d'une variable et la zone de la mémoire dans laquelle cette variable est stockée ? Reliez les cases correspondantes :

	◯ Données globales (*Global Data*)
Variable locale (*Automatic Storage Duration*) ◯	◯ Code machine (*Machine Code*)
Variable globale (*Static Storage Duration*) ◯	◯ Tas (*Heap Data*)
	◯ La pile (*Stack Data*)

Question 2 (2 points)

```
int main(int argc, char *argv[]) {
  int a = 4;
  int *b = &a;
  *b = 5;
  return EXIT_SUCCESS;
}
```

Juste avant la fin du programme (juste avant le `return`) :
Que contient la variable `a` ?

Que contient la variable `b` ?

Exercice 2 : Questions de cours (16 points / 26 minutes)

Nous développerons un simulateur simple de logique séquentielle, capable de simuler des circuits composés de portes NON (NOT) et OU exclusif (XOR), de bascules D (*D flip flop*) et de fils représentant les valeurs 0 ou 1.

Question 3 (2 points)

Pour distinguer le type des différents composants à simuler, on souhaite définir une énumération pouvant prendre les valeurs suivantes :

- `ZERO` : un fil représentant 0
- `ONE` : un fil représentant 1
- `NOT` : la sortie d'une porte NON
- `XOR` : la sortie d'une porte OU exclusif
- `REG` : la sortie d'une bascule D (D flip flop)

Donnez la définition de cette énumération et faites en sorte de pouvoir l'utiliser sous le type `kind_t` :

Question 4 (4 points)

On souhaite définir une structure regroupant les informations d'un composant à simuler et qui contient les membres suivants :

- Un membre `kind` qui représente le type du composant (voir `kind_t`).
- Deux membres de type booléen avec les noms `value` et `next_value`.
- Un membre `inputs` qui est un tableau de 2 pointeurs vers d'autres composants du circuit.

Donnez la définition de cette structure et faites en sorte de pouvoir l'utiliser sous le type `circuit_t` :

Question 5 (4 points)

Implémentez la fonction `step_combinational(circuit_t *c)` qui, en fonction du composant pointé par `c` :

- Pour un fil représentant 0, renvoie 0.
- Pour un fil représentant 1, renvoie 1.
- Pour une bascule D (D flip flop), renvoie sa valeur `value`.
- Pour une porte NON (NOT), appelle récursivement `step_combinational` sur sa première entrée (`inputs[0]`) puis renvoie l'opposé de cette valeur.
- Pour une porte OU exclusif (XOR), appelle récursivement `step_combinational` sur ses deux entrées (`inputs[0]` et `inputs[1]`) puis renvoie le XOR de ces deux valeurs.

Vous utiliserez de préférence une construction `switch` sur la valeur de la variable `c`.

```
// Evaluate the value of a circuit element
_Bool step_combinational(circuit_t *c) {
  // Provide your code below (using a switch!)...









}
```

Question 6 (2 points)

Fournissez les **arguments à la fonction** `printf` du code suivant, qui parcourt le tableau `c` de `n` composants d'un circuit. Pour tous les composants qui **représentent une bascule D**, la fonction est censée afficher les valeurs (voir `circuit_t`) de la bascule (`value`) à l'écran sous forme d'un unique **caractère**, soit le caractère `t` soit le caractère `f`.
Complétez le code suivant (à l'endroit des `___`) :

```
// Print the current value of each register on the screen:
void print_regs(circuit_t c[], unsigned int n) {
  for(unsigned int i = 0; i < n; i++) {
    if (c[i].kind == REG)
      printf("___", c[i].value ? ___ : ___);  // 3x answers at the ___
  }
  printf("\n");
}
```

Question 7 (4 points)

Complétez la fonction `main` en **allouant un tableau** sur le tas. Déterminez le **nombre d'éléments** que le tableau doit contenir en examinant le code. Vérifiez les erreurs. Affichez un **message d'erreur** explicite si une erreur survient. Enfin, n'oubliez pas de **libérer la mémoire** du tas avant de quitter le programme.

```
// Function to simulate a clock tick:
void step_circuit(circuit_t c[], unsigned int n) {
  // Compute next value of each register ...
  for(unsigned int i = 0; i < n; i++) {
    if (c[i].kind == REG)
      c[i].next_value = step_combinational(c[i].inputs[0]);
```

```
  }

  // Copy next value of each register into its value field.
  for(unsigned int i = 0; i < n; i++) {
    if (c[i].kind == REG)
      c[i].value = c[i].next_value;
  }
}

int main(int argc, char *argv[]) {
  // Provide your code below ...
  circuit_t *c =









  // Create 4 circuit elements for a simple 2-bit counter:
  circuit_t *not = &c[0], *xor = &c[1], *reg0 = &c[2], *reg1 = &c[3];
  mk_not(not , reg0);
  mk_xor(xor , reg0, reg1);
  mk_reg(reg0, not);
  mk_reg(reg1, xor);

  // Simulate the 4 circuit elements for 5 clock ticks:
  for(unsigned int i = 0; i < 5; i++) {
    print_regs(c, 4);
    step_circuit(c, 4);
  }

  // Provide your code below (if needed) ...









  return EXIT_SUCCESS;
}
```