1 Home Run (14 pts)

1. **(3 pts)** On considère le pseudo-code ci-dessous. Transformez le pour mettre en avant les blocs de base et respecter les contraintes des microprocesseurs. **Identifiez clairement le nombre de blocs de base alors obtenus.** *Rappel cjump <cond>, <iftrue>, <iffalse>.*

Réponse :

```
1     a ← 41
2     b ← [x] # Memory access
3     c ← 1
4  L1: cjump c ≠ 10, L7, L9
5  L2: jump L1
6  L3: cjump b > 0, L2, L6
7  L4: b ← d
8  L5: jump L2
9  L6: cjump b < 0, L2, L5
10 L7: x ← x + 4
11    c ← c + 1
12    d ← [x]
13    cjump b ≠ d, L4, L8
14 L8: cjump b = 0, L2, L3
15 L9: a ← a + 1
16    return # a and b are liveout
```

2. **(1 pts)** Dessinez le graphe de flot de contrôle de l'exemple ci dessus, c'est à dire l'exemple **sans vos modifications de la question (1) !** *Vous utiliserez ℓ_i pour indiquer la ligne i.*

Réponse :

3. **(4 pts)** Pour chaque nœud, indiquez les variables qui sont définies, celles qui sont utilisées, celles live_in et celles qui sont live_out. Pour rappel :

$$in[n] = use[n] \cup (out[n] \backslash def[n]) \qquad out[n] = \bigcup_{s \in succ(n)} (in[s])$$

Réponse :

	def	use	live_in	live_out
ℓ_1				
ℓ_2				
ℓ_3				
ℓ_4				
ℓ_5				
ℓ_6				
ℓ_7				
ℓ_8				
ℓ_9				
ℓ_{10}				
ℓ_{11}				
ℓ_{12}				
ℓ_{13}				
ℓ_{14}				
ℓ_{15}				
ℓ_{16}				

4. **(1 pts)** Re-dessinez le graphe de flot de contrôle de la question (2) en y faisant apparaître la vivacité de chaque variable.

Réponse :

5. **(1 pts)** Déduisez en le graphe d'interférence du programme.

Réponse :

6. **(1 pts)** Calculez la *spill priority* des variables a, b, c, d et x. Pour rappel :

$$spill\_priority = ((use\_outside\_loop + def\_outside\_loop) + (10 * use\_inside\_loop + def\_inside\_loop))/degree$$

Réponse :

7. **(2 pts)** Réécrivez le code en effectuant un spill de *a* à l'adresse [sp+4]. *Les lignes ci-dessous sont là pour vous aider à écrire droit, il peut y en avoir plus que nécessaire.*

Réponse :

ℓ_1		ℓ_{11}	
ℓ_2		ℓ_{12}	
ℓ_3		ℓ_{13}	
ℓ_4		ℓ_{14}	
ℓ_5		ℓ_{15}	
ℓ_6		ℓ_{16}	
ℓ_7		ℓ_{17}	
ℓ_8		ℓ_{18}	
ℓ_9		ℓ_{19}	
ℓ_{10}		ℓ_{20}	

8. **(1 pts)** En ignorant la variable a, que fait le programme de la question 1 ?

Réponse :

1. **(1 pt)** À quoi sert le déroulage de boucle ? Vous donnerez un exemple haut niveau (code C) pour expliquer votre propos.

Réponse :

3. **(1 pts)** Qu'est ce que la linéarisation ? **Linéarisez le code suivant.**

```
1  t := 1
2  call(foo,
3       t + (t:= 51, 0),
4       12, 42)
```

Réponse :

4. **(1 pt)** Expliquez le rôle du middle-end dans un compilateur.

Réponse :

5. **(1 pts)** En considérant l'imbrication des fonctions suivantes, vous préciserez l'état de la pile à la fin de l'éxécution de la ligne 5 (suite à l'appel de la ligne 13). Vous préciserez les **static link** et **indiquerez** leur rôle.

```
1  void one() {
2    int x1 = 1;
3
4    void two() {
5      int x2 = 2;
6    }
7
8    void three() {
9      int x3 = 3;
10     two();
11   }
12
13   three();
14 }
```

Réponse :

6. **(1 pt)** Quelles sont les trois traductions possibles pour l'expression $\alpha > \beta$? Il ne vous est pas demandé du code mais uniquement l'explication des trois cas.

Réponse :