Exercice 1 : logique de Hoare (*5 points*)

Pour chacun des triplets de Hoare suivants, indiquer s'il est valide ou non. Si le triplet n'est pas valide, donner un état qui satisfait la précondition mais ne satisfait pas la postcondition. Si le triplet est valide, en donner une démonstration en utilisant les règles de la logique de Hoare.

Question 1 : $\{x = -y \wedge x \neq n\}\ x := x + 1;\ y := y - 1\ \{x = -y\}$

(0,5 point) Ce triplet est valide.
Une démonstration de ce triplet en utilisant les règles de la logique de Hoare est :
(1 point)

$$\dfrac{(1) \qquad \dfrac{}{\{x = -y + 1\}\ y := y - 1\ \{x = -y\}}\ (\textit{assignment})}{\{x = -y \wedge x \neq n\}\ x := x + 1;\ y := y - 1\ \{x = -y\}}\ (\textit{sequence})$$

(0,5 point) où (1) est

$$\dfrac{x = -y \wedge x \neq n \Rightarrow x + 1 = -y + 1 \qquad \dfrac{}{\{x + 1 = -y + 1\}\ x := x + 1\ \{x = -y + 1\}}\ (\textit{assignment})}{\{x = -y \wedge x \neq n\}\ x := x + 1\ \{x = -y + 1\}}\ (\textit{precondition})$$

L'implication $x = -y \wedge x \neq n \Rightarrow x + 1 = -y + 1$ résulte de $A \wedge B \Rightarrow A$ et de $x = y \Rightarrow x + 1 = -y + 1$.

Question 2 : $\{z = 0\}$ while $z = z$ do $z := z + 1$ od $\{z = 2\}$

(0,5 point) Ce triplet est valide.
Une démonstration de ce triplet en utilisant les règles de la logique de Hoare est :
(0,5 point)

$$\dfrac{z = 0 \Rightarrow z \geq 0 \quad \{z \geq 0\}\ \text{while } z = z \text{ do } z := z + 1 \text{ od } \{z \neq z \wedge z \geq 0\} \quad z \neq z \wedge z \geq 0 \Rightarrow z = 2}{\{z = 0\}\ \text{while } z = z \text{ do } z := z + 1 \text{ od } \{z = 2\}}$$

selon la règle (*consequence*).
(0,5 point) La première prémisse $z = 0 \Rightarrow z \geq 0$ est évidemment vraie.
(0,5 point) La troisième prémisse est vraie car $z \neq z$ est faux et à partir du faux on peut tout déduire.
(1 point : 0,5 pour (*while*) et 0,5 pour (*precondition*)) La deuxième prémisse se démontre par

$$\dfrac{\{z = z \wedge z \geq 0\}\ z := z + 1\ \{z \geq 0\}}{\{z \geq 0\}\ \text{while } z = z \text{ do } z := z + 1 \text{ od } \{z \neq z \wedge z \geq 0\}}\ (\textit{while})$$

dont la prémisse se démontre par

$$\dfrac{z = z \wedge z \geq 0 \Rightarrow z + 1 \geq 0 \quad \{z + 1 \geq 0\}\ z := z + 1\ \{z \geq 0\}}{\{z = z \wedge z \geq 0\}\ z := z + 1\ \{z \geq 0\}}\ (\textit{precondition})$$

L'implication $z = z \wedge z \geq 0 \Rightarrow z + 1 \geq 0$ est évidente. L'instance

$$\dfrac{}{\{z + 1 \geq 0\}\ z := z + 1\ \{z \geq 0\}}$$

de la règle (*assignment*) démontre $\{z + 1 \geq 0\}\ z := z + 1\ \{z \geq 0\}$.

Exercice 2 : division euclidienne par soustractions (*9 points*)

Dans cet exercice, on considère une logique de Hoare dont toutes les constantes et toutes les variables sont des entiers relatifs (nombres positifs ou négatifs), et qui admet une addition, une soustraction binaire, une multiplication et des comparaisons ($\leq$, $<$, $\geq$, $>$ et $=$) entre deux entiers relatifs.

Question 1 : Exprimer par un triplet de Hoare que le programme

$$q := 0;\ r := a;\ \textsf{while}\ r \geq b\ \textsf{do}\ q := q + 1;\ r := r - b\ \textsf{od}$$

calcule dans q et r le quotient et le reste de la division euclidienne de a par b, pour tous les entiers naturels a et pour tout entier naturel b non nul. La postcondition de ce triplet doit relier a, b, q et r par une égalité et fixer les bornes les plus précises possibles pour r.

(1 point) $\{a \geq 0 \wedge b > 0\}\ q := 0;\ r := a;\ \textsf{while}\ r \geq b\ \textsf{do}\ q := q+1;\ r := r-b\ \textsf{od}\ \{a = bq+r \wedge 0 \leq r < b\}$
La précondition $a \geq 0$ est requise pour pouvoir démontrer la postcondition $0 \leq r$.

Question 2 : Dans un tableau, appliquer à ce triplet de Hoare les règles (*sequence*), (*assignment*) et (*precondition*) de la logique de Hoare, jusqu'à ce qu'il ne reste plus qu'à démontrer un triplet de la forme $\{\ldots\}$ while ... do ... od $\{\ldots\}$.

(2,5 points : 0,5 point par ligne sauf 2 et 5) Dans la logique de Hoare, une démonstration de validité de ce triplet de Hoare est :

N°	Triplet ou formule	Justification
1	$\{a \geq 0 \wedge b > 0 \wedge q = 0 \wedge a = a\}\ r := a\ \{a \geq 0 \wedge b > 0 \wedge q = 0 \wedge r = a\}$	(*assignment*)
2	$\{a \geq 0 \wedge b > 0 \wedge q = 0 \wedge r = a\}$ while $r \geq b$ do $q := q + 1;\ r := r - b$ od $\{a = bq + r \wedge 0 \leq r < b\}$	
3	$\{a \geq 0 \wedge b > 0 \wedge 0 = 0 \wedge a = a\}\ q := 0\ \{a \geq 0 \wedge b > 0 \wedge q = 0 \wedge a = a\}$	(*assignment*)
4	$\{a \geq 0 \wedge b > 0 \wedge q = 0 \wedge a = a\}\ r := a;$ while $r \geq b$ do $q := q + 1;\ r := r - b$ od $\{a = bq + r \wedge 0 \leq r < b\}$	(*seq.*) 1 2
5	$a \geq 0 \wedge b > 0 \Rightarrow a \geq 0 \wedge b > 0 \wedge 0 = 0 \wedge a = a$	évident
6	$\{a \geq 0 \wedge b > 0 \wedge 0 = 0 \wedge a = a\}\ q := 0;\ r := a;$ while $r \geq b$ do $q := q + 1;\ r := r - b$ od $\{a = bq + r \wedge 0 \leq r < b\}$	(*seq.*) 3 4
7	$\{a \geq 0 \wedge b > 0\}\ q := 0;\ r := a;$ while $r \geq b$ do $q := q + 1;\ r := r - b$ od $\{a = bq + r \wedge 0 \leq r < b\}$	(*pre.*) 5 6

Question 3 : Proposer un invariant pour la boucle while, assez précis pour permettre de démontrer la validité de ce triplet de Hoare.

(1 point) La formule $a = bq + r \wedge r \geq 0$ est un invariant de boucle suffisant pour démontrer la validité de ce triplet de Hoare.

Question 4 : Donner les trois prémisses qu'on obtient en appliquant les règles (*consequence*) et (*while*), avec cet invariant, au triplet de la forme $\{\ldots\}$ while ... do ... od $\{\ldots\}$ obtenu précédemment.

(1,5 points : 0,5 point par formule) Pour démontrer que ce triplet de Hoare est valide, on applique les règles (*consequence*) et (*while*) avec cet invariant, et on obtient les trois prémisses suivantes :

$$a \geq 0 \wedge b > 0 \wedge q = 0 \wedge r = a \Rightarrow a = bq + r \wedge r \geq 0 \quad (1)$$

$$\{r \geq b \wedge a = bq + r \wedge r \geq 0\}\ q := q + 1;\ r := r - b\ \{a = bq + r \wedge r \geq 0\} \quad (2)$$

$$\neg(r \geq b) \wedge a = bq + r \wedge r \geq 0 \Rightarrow a = bq + r \wedge 0 \leq r < b \quad (3)$$

Question 5 : En appliquant une par une les règles de la logique de Hoare, démontrer la deuxième de ces prémisses, qui est le triplet de Hoare qui exprime que l'invariant est préservé par le corps de la boucle.

(3 points : 1 point pour (*sequence*) + 0,5 points pour chaque autre ligne) Dans la logique de Hoare, une démonstration de validité du triplet de Hoare

$$\{r \geq b \wedge a = bq + r \wedge r \geq 0\}\ q := q + 1;\ r := r - b\ \{a = bq + r \wedge r \geq 0\}$$

est :

N°	Triplet ou formule	Justification
1	$r \geq b \wedge a = bq + r \wedge r \geq 0 \Rightarrow a = b(q+1) + r - b \wedge r - b \geq 0$	évident
2	$\{a = b(q+1) + r - b \wedge r - b \geq 0\}\ q := q + 1\ \{a = bq + r - b \wedge r - b \geq 0\}$	(*assignment*)
8	$\{r \geq b \wedge a = bq + r \wedge r \geq 0\}\ q := q + 1\ \{a = bq + r - b \wedge r - b \geq 0\}$	(*precondition*) 1 2
9	$\{a = bq + r - b \wedge r - b \geq 0\}\ r := r - b\ \{a = bq + r \wedge r \geq 0\}$	(*assignment*)
10	$\{r \geq b \wedge a = bq + r \wedge r \geq 0\}\ q := q + 1;\ r := r - b\ \{a = bq + r \wedge r \geq 0\}$	(*sequence*) 8 9

Exercice 3 : inversion d'un tableau dans un autre tableau, en micro-C (*6 points*)

Question 1 : Reproduire le code suivant et compléter les pointillés pour que son exécution stocke les éléments du tableau `tab`, supposé de taille `n`, dans le tableau `out`, mais dans l'ordre inverse, sans modifier le contenu du tableau `tab`.

Le tableau `out` est supposé être disjoint du tableau `tab`, alloué avant cette exécution, et de taille suffisante. Par exemple, si le tableau `tab` est | 4 | 1 | 3 | 3 |, alors, après l'exécution de l'instruction

```
inversion(tab,out);
```

le début du tableau `out` doit être | 3 | 3 | 1 | 4 |.

```
void inversion (int tab[], int n, int out[]) {
  /*@ requires length(tab) == n;

  */
  int i = 0;
  while (i < ..) {
```

```
    /*@

    */
    out[..] = tab[n-1-i];
    i = ..;
  }
}
```

(1,5 points) voir la réponse et le barème détaillé dans le listing suivant.

Question 2 : Donner en syntaxe micro-C une précondition pour que le tableau `out` ait une taille suffisante pour contenir les éléments du tableau `tab` dans l'ordre inverse.

(0,5 points) voir la ligne 2 du listing suivant.

Question 3 : Donner en syntaxe micro-C une postcondition qui exprime que le tableau `out` contient les éléments du tableau `tab` dans l'ordre inverse.

(1 point) voir la ligne 3 du listing suivant.

Question 4 : Proposer en syntaxe micro-C un invariant de boucle suffisant pour démontrer cette postcondition.

(2 points) voir les lignes 6 et 7 du listing suivant.

Question 5 : Proposer en syntaxe micro-C un variant de boucle permettant de démontrer la terminaison de la boucle et de la fonction.

(1 point) voir la ligne 8 du listing suivant.

```
1  void inversion (int tab[], int n, int out[]) {
2    /*@ requires length(tab) == n <= length(out);                // Q2: 0.5 pt
3        ensures forall i. 0 <= i < n -> out[i] == tab[n-1-i]; */ // Q3: 1 pt
4    int i = 0;
5    while (i < n) {                                              // Q1: 0.5 pt
6      /*@ invariant 0 <= i <= n && forall j. 0 <= j < i
7            -> out[j] == tab[length(tab)-1-j];                   // Q4: 2 pts
8          variant n-i; */                                        // Q5: 1 pt
9      out[i] = tab[n-1-i];                                       // Q1: 0.5 pt
10     i = i+1;                                                   // Q1: 0.5 pt
11   }
12 }
```