1. Considera o sistema $\mathcal{H}_g$ (o sistema de inferência para a lógica de Hoare sem a regra da consequência) e as funções VCG, VC e wp apresentadas nas aulas.
   (a) Aplica o algoritmo VCGen definido nas aulas para calcular as condições de verificação do seguinte programa (acrescenta o correspondente invariante $I$):

```
{y=i, y>=0}
 z:=0;
 while y!= 0 do {I}
   z:=z + x;
   y:=y-1
{z=x*i}
```

   (b) Adapta o sistema $\mathcal{H}_g$ para um que permita a correção total de triplos de Hoare (para programas da linguagem **while**).

2. Considera um programa que calcula a divisão inteira de $x$ por $y$ (inteiros), guardando o quociente em $z$ e o resto em $w$.
   (a) Escreve especificações ACSL para o programa acima.
   (b) Escreve uma função em **C** que satisfaça as especificações indicadas na alínea anterior e usa Frama-C-jessie para verificar a sua correcção.

3. Considera a ferramenta Frama-C/jessie (para geração de condições de verificação) e alguns demonstradores automáticos (pelo menos Simplify e Alt-ergo)
   (a) Escrever uma função **C** que dado um array de inteiros o transforme noutro por aplicação duma transformação há tua escolha que não a identidade (pode ser alterado o próprio ou criado um novo).
   (b) Especifica as pré-condições necessárias para a verificação de segurança dessa função. Gera as condições de verificação (com possiveis auxiliares lógicos), testa quais os demonstradores automáticos que as verificam. Deves guardar esta informação num directório com nome terminado em 0.
   (c) Especifica as pós-condições que garantam a correção da função em relação ao enunciado dado (verificações funcionais). Especifica também os invariantes e variantes de ciclo necessários (e eventuais guardas). Gera as condições de verificação (com possíveis auxiliares lógicos), testa quais os demonstradores automáticos que as verificam. Deves guardar esta informação num directório com nome terminado em 1.

4. Considera o tipo de dados abstracto `Queue` para modelar filas de inteiros em **C**.
   (a) Escreva protótipos e contractos para as funções usuais sobre pilhas `isEmpty`, (retorna uma fila vazia), `enqueue`, `dequeue`, e `size`. Os contractos devem ser válidos para qualquer implementação concreta de `Queue`.
   (b) Considera agora a definição concreta `typedef int Queue [SIZE]`. Define em **C** implementações ACSL-anotadas para as funções da alínea anterior.
   (c) Mostra a correcção de cada uma das funções implementadas, relativamente às especificações definidas anteriormente.