universite PARIS-SACLAY

**Master 1 Informatique MPRI**
**Deductive program verification**
**2018 / 2019**

# Exam : Deductive verification

Name(s) :
Group :
Date :

**Exercise 1** We consider the $\mu$ML with pointers.
Let $P$ and $Pre$, the following program and precondition :

$$P \triangleq q = p; \star r = \star p; \star q = \star p + 1$$
$$Pre \triangleq p \neq q \land p \neq r \land q \neq r \land \text{read } mem\ p = 0$$

Which Hoare triples are valid ?

1. $\{Pre\}\ P\ \{p = q\}$
2. $\{Pre\}\ P\ \{\text{read } mem\ p = \text{read } mem\ q\}$
3. $\{Pre\}\ P\ \{p = r\}$
4. $\{Pre\}\ P\ \{\text{read } mem\ p = \text{read } mem\ r\}$
5. $\{Pre\}\ P\ \{\text{read } mem\ q = 1 \land \text{read } mem\ r = 1\}$
6. $\{Pre\}\ P\ \{\text{read } mem\ q = 1 \land \text{read } mem\ r = 0\}$

**Exercise 2** The following function inverts an array of integers.

```
let invert (a: array int) : unit
  ensures { forall i. 0 <= i < length a -> a[i] = old a[length a - i - 1] }
=
  let ref k = 0 in
  while k < length a do
    invariant (* 1 *) { 0 <= k <= length a }
    invariant (* 2 *) { forall i. 0 <= i < k -> a[i] = old a[length a - i - 1] }
    invariant (* 3 *) { forall i. 0 <= i < k -> a[length a - i - 1] = old a[i] }
    variant   { length a - k }
    let x = a[length a - k - 1] in
    a[length a - k - 1] <- a[k];
    a[k] <- x;
    k <- k + 1
  done
```

For each of the following statements, determine whether they are true or false.

1. The invariant 1 is true when the loop starts.
2. The invariant 2 is true when the loop starts.
3. The invariant 3 is true when the loop starts.

4. The invariant 1 is preserved by the loop.
5. The invariant 2 is preserved by the loop.
6. The invariant 3 is preserved by the loop.
7. The invariants are sufficient to prove the postcondition.
8. The variant is sufficient to prove the termination of the loop.

How should we fix the code so that it conforms to the specification “inverts the given array” ?

Once the code is fixed, what should we modify in the code annotations so that all above statements are true ?

**Exercise 3** We consider this code :

```
exception Invalid

let pivot (a: array int) (v: int) : unit =
  let ref l = 0 in
  let ref u = length a - 1 in
  while l <= u do
    if a[l] <= v then l <- l + 1 else
    if a[u] >= v then u <- u - 1 else
    raise Invalid
  done
```

Give the complete contract of `pivot`, with loop invariant and variant.

**Exercise 4** The WhyML function `search` searches the given value in a sorted array :

```
exception Found int

let search (a: array int) (v: int) : unit
  requires { forall i j. 0 <= i < j < length a -> a[i] <= a[j] }
  ensures  { forall k.  0 <= k < length a -> a[k] <> v }
  raises   { Found k -> 0 <= k < length a /\ a[k] = v }
=
  let ref l = 0 in
  let ref u = length a - 1 in
  while l <= u do
    invariant { 0 <= l /\ u < length a }
    invariant { forall k. 0 <= k < length a -> k < l \/ u < k -> a[k] <> v }
    let m = (l + u) / 2 in
    if v < a[m] then l <- m + 1 else
    if v > a[m] then u <- m - 1 else
    raise (Found m)
  done
```

Using weakest precondition calculus for partial correction, give the verification condition of the function `search`.

You can name the precondition $P(a)$, the normal postcondition $Q(a,v)$, the postcondition related to the exception $E(a,v,k)$, et the invariants $I_1(a,l,u)$ and $I_2(a,v,l,u)$.

The `search` function cannot be proved with the provided specification.

Assuming that the specification is correct, propose a correct implmentation.

Assuming that the code is correct, fix the specification.

**Exercise 5** Consider the following C function, from the libc.

```
void *memchr(const void s[.n], int c, size_t n);

The  memchr()  function  scans  the  initial n bytes of the memory area
pointed to by s for the first instance of c.  Both c and the  bytes  of
the memory area pointed to by s are interpreted as unsigned char.
```

1. Replace `\false` in the following definition of the ACSL predicate `mem` such that the predicate is true if and only if the integer `c` is one of the `n` first characters of `s`.

   ```
   /*@ predicate mem(unsigned char *s, integer c, integer n) =
         \false;
     */
   ```

2. Using the previous question and using ACSL behaviors, give a specification for the `memchr` function.

**Exercise 6** We consider a 8 bits architecture where values of type `int` are in range $-128$ to 127. Let `p` and `x`, two C variables of type `int**` and `int`.

Give a list of ACSL assertions that guarantees, if all of them are verified, that the following C expression does not create any undefined behaviors. Justify informally the behavior that each of the assertion avoids.

$$e \triangleq -\texttt{(*(*(p+1)+2)/x)}$$

**Exercise 7** We modify the $\mu$ML language to introduce the assignment $x \leftarrow t$ (where $x$ is a variable and $t$ a term) not as an expression but as a term. The type of this term is not *unit* anymore but the same as $t$. Its semantics also changes : now, $x \leftarrow t$ modifies the value of $x$ with the value of $t$ (like in the lecture), **and then returns this value**.

For example, allowing assignments in terms allows writing the following term which does not belong to our original $\mu$ML :

$$x \leftarrow 1 + (y \leftarrow 2) + 3.$$

Its semantics is to write 2 in $y$ and 6 in $x$, and finally return 6.

1. Define inductively the operator $\phi$ that takes in input the term $t$ in this new language and returns a $\mu$ML expression that produces the same effects as $t$ before returning its result.

   We can limit ourselves to assignment, unary and binary operators and bound as many local variables $tmp_1, tmp_2, \ldots$ as necessary.
2. Is there only one way to define $\phi$ ? If so, explain why. Else, give an example in which we could obtain different results depending on how $\phi$ is defined.
3. Using question 1, give a WP rule for $x \leftarrow t$ in this new language.