Innopolis University | Software Architecture | Spring 2016

Weakest PreCondition Calculus

Néstor Cataño | **Extra Points – Week 11** | Issued: 28/March/2016 Due: 28/March/2016

Questions

Use **only** this sheet to write your answers. No additional sheets will be provided. Answers in other sheets will not be accepted. Write your name on the top part of this sheet.

1. (**5 marks**) Use WP Calculus to check if the following program is correct or not. The requires comment part is `setAge` method pre-condition, and the ensures part is is `setAge` method post-condition.

```
/*@ requires a >= 0;
    ensures age >= 0;
 */
void setAge(int a) {
 age = a;
}
```

   The program is correct if and only if `a >= 0` implies `WP(age = a, age >= 0)`. The weakest pre-condition reduces to `a >= 0` after replacing `age` by `a` in `age >= 0`. `a >= 0` implies `a >= 0` is a tautology so the program is correct.

2. (**5 marks**) The program below is written in pseudocode. What is the WP necessary to establish the property `answer == even`? Why? Justify your answer.

```
case i mod 2 of
 either 0 then answer := even end
 or 1 then answer := odd end
end
```

   We need to calculate `WP(prog, answer == even)`, where `prog` is the code given above. This code works just like a standard if-then-else statement in which the guard is "`i mod 2 == 0`". Hence, by applying the if-then-else Hoare rule to `prog`, the weakest precondition reduces to the following expression.

```
i mod 2 == 0 => WP(answer := even, answer == even) &&
i mod 2 == 1 => WP(answer := odd, answer == even)
```

   The two conjuncts above reduce to the following expressions:

```
i mod 2 == 0 => even == even &&
i mod 2 == 1 => odd == even
```

The first conjunct reduces to true and the second conjunct reduces to the negation of `i mod 2 == 1`, that is, it reduces to `i mod 2 == 0`. Therefore, the weakest precondition for program `prog` to achieve `answer == even` is `i mod 2 == 0`.

3. (**5 marks**) Use WP calculus to determine if the following Hoare triple is correct or not. Assume that both `x` and `y` are positive natural numbers. The symbol ÷ stands for division of natural numbers, for which there is no *remainder*.

   `{x > 0} y:= y÷x {y >= 1}`

   The program is correct if and only if `x > 0` implies `WP(y:= y÷x, y >= 1)`. The WP expression reduces to `y÷x >= 1`. This reduces to `y >= x`. This last expression is not a tautology, therefore, the original Hoare triple is incorrect.