# Exercises 1 - solutions

February 19, 2013

## Find the invariant 1

```
class Mult {
  int fi(int x, int y)
    requires x >= 0 && y >= 0;
    ensures result == x * y;
  {
    int r = 0;
    int i = 0;
    while (i < y)
      invariant r == x * i && i <= y;
    {
      i = i + 1;
      r = r + x;
    }
    return r;
  }
}
```

## Find the invariant 2

```
class Mult {
  int fi2(int x, int y)
    requires x >= 0 && y >= 0;
    ensures result == x * y;
  {
    int r = 0;
    int i = y;
    while (i > 0)
      invariant r == (y - i) * x && i >= 0;
    {
      i = i - 1;
      r = r + x;
    }
    return r;
  }
}
```

## What's missing?

```
  class Account {
    private int checkingBalance;
    private int savingsBalance;

    private void checkingDeposit(int amount)
-->   requires amount > 0;
      ensures checkingBalance == old(checkingBalance) + amount;
    {
      checkingBalance = checkingBalance + amount;
    }

    private void checkingWithdraw(int amount)
-->   requires amount > 0 && checkingBalance >= amount;
      ensures checkingBalance == old(checkingBalance) - amount;
    {
      checkingBalance = checkingBalance - amount;
    }

    private void savingsDeposit(int amount)
-->   requires amount > 0;
      ensures savingsBalance == old(savingsBalance) + amount;
    {
      savingsBalance = savingsBalance + amount;
    }
    ...
  }
```

## Why does this fail to verify?

Now add the following two methods to the class above:

```
  private void transferSavingsToChecking(int amount)
    requires amount > 0 && savingsBalance >= amount;
    ensures savingsBalance == old(savingsBalance) - amount;
    ensures checkingBalance == old(checkingBalance) + amount;
    ensures old(checkingBalance) + old(savingsBalance) == checkingBalance + savingsBalance;
  {
    savingsBalance = savingsBalance - amount;
    checkingBalance = checkingBalance + amount;
  }

  private void transferCheckingToSavings(int amount)
    requires amount > 0 && checkingBalance >= amount;
    ensures old(checkingBalance) + old(savingsBalance) == checkingBalance + savingsBalance;
  {
    checkingWithdraw(amount);
    savingsDeposit(amount);
  }
```

Solution: the methods checkingWithdraw and savingsDeposit need modifies clauses, since the default is "modify everything".

## Find the invariant 3

```
class Sqrt {
  public int sqrt(int x)
    requires 0 <= x;
    ensures result*result <= x && x < (result+1)*(result+1);
  {
    int r = 0;
    while ((r+1)*(r+1) <= x)
      invariant r*r <= x;
    {
      r++;
    }
    return r;
  }
}
```

## Find the invariant 4

```
class LinearSearch {
  bool search(int[] a, int key)
    ensures result == exists{int i in (0: a.Length); a[i] == key};
  {
    int n = a.Length;
    do
      invariant 0 <= n && n <= a.Length;
      invariant forall{int i in (n: a.Length); a[i] != key};
    {
      n--;
      if (n < 0) {
        break;
      }
    } while (a[n] != key);
    return 0 <= n;
  }
}
```

## Find the error

```
class SomeFunction1 {
  public int f(int x, int y)
    requires 0 <= x && 0 <= y;
    ensures result == 2*x + y;    // x + y
  {
    int r = x;
    for (int n = 0; n < y; n++)
      invariant r == x+n && n <= y;
    {
      r++;
    }
    return r;
  }
}
```

## Find the invariant - last one

```
class GCD {
  int gcd(int a, int b)
    requires a > 0 && b > 0;
    ensures result > 0 && a % result == 0 && b % result == 0;
    ensures
      forall{int k in (1..a+b), a % k ==0 && b % k == 0; k <= result};
  {
    int i = 1; int res = 1;
    while (i < a+b)
      invariant i <= a+b;
      invariant res > 0 && a % res == 0 && b % res == 0;
      invariant
        forall{int k in (1..i), a % k == 0 && b % k == 0; k <= res};
    {
      i++;
      if (a % i == 0 && b % i == 0) {
        res = i;
      }
    }
    return res;
  }
}
```