Mid-conditions: Tutorial sheet 4

Jeremy Bradley

31 January 2005

PMT Exercise 10: Questions **1(c) and 3** are assessed and are due in to the SAO by 4.30pm on **8 February 2005**. This is a hardcopy submission but you still need to register your submission using CATE which will also provide you with your submission cover sheet: `https://sparrow.doc.ic.ac.uk/~cate/`

1. For the `intMax` program below:

```
    int intMax(int x, int y) {
        // pre: none
        // post: (res == x_0 || res == y_0)
        //          && (res >= x_0 && res >= y_0))

        int res;
        if (x >= y)
[1]           res = x;
        // mid (a): (res == x_0) && (res >= y_0)
        else
[2]           res = y;
        // mid (b): (res == y_0) && (res >= x_0)
        return res;
    }
```

   Construct the following:

   (a) The combined mid-condition

   (b) A natural deduction proof from the pre-condition to the combined mid-condition

   (c) (**Assessed**) A natural deduction proof from mid-condition (b) to the post-condition

2. For the method `weird`, a specification would take the form: $\vdash x = 7$

```
   int weird (int x) {
       // pre: none
       // post: x == 7
[1]    x = x * 2;
[2]    x = 7;
       return x;
   }
```

Using extra variables where necessary, prove the specification is satisfied by the method.

[**continued**...

3. (**Assessed**) As in the notes, the class `Point` is defined with methods `up` and `right` as follows.

```
class Point {
    int xc;
    int yc;

    Point (int i, int j) {
        xc = i;
        yc = j;
    }

    public void up (int n) {
        // Pre: none
        // Post: xc == xc_0 && yc == yc_0 + n
        yc = yc + n;
    }
    public void right (int n) {
        // Pre: none
        // Post: xc == xc_0 + n && yc == yc_0
        xc = xc + n;
    }

}

class Square {
    public static upleft (Point P, int n) {
        // Pre: none
        // Post: xc == xc_0 - n && yc == yc_0 + n

[1]     P.up(n);
[2]     P.right(-n);
    }
}
```

Assuming that the post-conditions of `Point.up` and `Point.right` are true. Show, by natural deduction, that the pre-condition entails the post-condition for the method, `Square.upleft`.

4. Given the following definition of a `Sphere` class which also calculates the volume and surface area:

```
class Sphere {
    int r; \\ sphere radius

    double volume () {
        // Pre: r_0 >= 0
        // Post: res == 4/3 * PI * r_0^3 && r == r_0
[1]     double res = r / 3;
[2]     res = res * this.area();
        return res;
    }

    double area () {
        // Pre: r_0 >= 0
        // Post: res == 4 * PI * r_0^2 && r == r_0
        double res = 4 * PI * r * r;
        return res;
    }
}
```

(a) State the specification of `Sphere.volume` from the pre- and post-condition comments

(b) Use natural deduction to justify that `Sphere.volume` meets its specification, given that `Sphere.area` has been validated.