Automated Testing and Verification

Exercises – Program verification with ESC/Java2

Exercise 1

Add *requires* annotations such that method *sum* does not signal an exception.

```
//@ requires ...
int sum(int array[], int len) {
  int sum = 0;
  int i = 0;
  while (i < len) {
    sum = sum + array[i];
    i = i + 1;
  }
  return sum;
}
```

Exercise 2

Add annotations *requires*, *loop_invariant* and *decreasing* such that the *ensures* annotation holds:

```
//@ requires ...
//@ ensures \result == m*n;
int multiply(int m, int n) {
  int product = 0;
  int i = 0;
  //@ loop_invariant ...
  //@ decreasing ...
  while (i < n) {
    product = add(product, m);
    i = i + 1;
  }
  return product;
}
```

Exercise 3

Add a frame condition to the following class:

```
class C {
  Object /*@ ...@*/ object;

  //@ modifies ...
  void changeObject() {
    if (this.object==null)
      this.object = null;
    else
      this.object = this.objet;
  }
}
```

Which annotation will you consider for field `object`: *nullable* or *non_null*?

Exercise 4

Why specifications for methods *hashCode_1* and *hashCode_2* are different?

```
public int hashCode_1(Object o) {
  // @ asume o!=null;
  return o.hashCode():
}

public int hashCode_2(Object o) {
  // @ assert o!=null;
  return o.hashCode():
}
```

Exercise 5

Add annotations to the following class:

```
class StringAppender {
  String str = "";

  // @ normal_behavior ...
  // @ also
  // @ exceptional_behavior ...
  public String append(Object o) throws ConcatException {
    if (o!=null) {
      str += o.toString();
    } else {
      throw new  ConcatException("argument cannot be null");
    }
  }
}
```

Exercise 6

Install static verifier ESC/Java2 (`http://secure.ucd.ie/products/opensource/ESCJava2/`). Modify the annotations until the static verifier signals no more warnings. Please remember to add the *-LoopSafe* options when running ESC/Java2.

Exercise 7

Run the ESC/Java2 static verifier over the following class:

```
class Bag {
  int[] a;
  int n;


  Bag(int[] input) {
    n = input.length;
    a = new int[n];
    System.arraycopy(input, 0, a, 0, n);
  }
```

```
  int extractMin() {
    int mindex=0;
    int m=a[mindex];
    int i=1;
    for (i=1;i<n;i++) {
      if (a[i]<m) {
        mindex=i;
        m = a[i];
      }
    }
    n--;
    a[mindex]=a[n];
    return m;
  }


}
}
```

a) How many warnings are reported?
b) Add annotations until no more warnings are reported.