Example EXAM
Testing, Debugging, and Verification
TDA567

Extra aid:	Only dictionaries may be used. Other aids are *not* allowed!
Grade intervals:	**U**: 0 – 23p, **3**: 24 – 35p, **4**: 36 – 47p, **5**: 48 – 60p, **G**: 24 – 47p, **VG**: 48 – 60p, **Max.** 60p.

Please observe the following:

- This exam has **??** numbered pages.
  Please check immediately that your copy is complete
- Answers must be given in English
- Use page numbering on your pages
- Start every assignment on a fresh page
- Write clearly; unreadable = wrong!
- Fewer points are given for unnecessarily complicated solutions
- Indicate clearly when you make assumptions that are not given in the assignment

Good luck!

Assignment 1 (Testing) (12p)

(a) **Briefly** describe the main features of the *Extreme Testing* methodology.

(b) Construct a minimal set of test-cases for the code snippet below, which satisfy *condition coverage*

```
if (a > b || b < 0)
   return a;
else
   return b;
```

(c) We discussed another two criteria for logic coverage in class. Describe these. Also describe the relationship between the three logic based criteria. Does your test-set from (b) satisfy any of the other coverage criteria. Why/why not?

Assignment 2 (Debugging) (12p)

(a) When is a statement B control dependent on a statement A?

(b) In the below small Dafny program, on which statement(s) is/are the statements in line 9 data dependent?

```
1 method M(n : nat) returns (b : nat){
2       if(n == 0)
3          { return 0; }
4       var i := 1;
5       var a := 0;
6       b := 1;
7       while (i < n)
8       {
9        a, b := b, a+b;
11       i := i +1;
12      }
13 }
```

(c) On which statements is line 11 *backward dependent*?

(d) The `ddMin` algorithm computes a minimal failure inducing input sequence. It relies on having a method `test(i)` which returns `PASS` if the input `i` passes the test or `FAIL` if the `i` causes failure (i.e. bug is exhibited).
Suppose our input consists of representations of DNA sequences, made out out the letters `G,A,T,C` which represent the nucleotides. Let `test` return `FAIL` whenever the DNA sequence contains *two consecutive occurrences of the letter* C somewhere in the sequence.
Simulate a run of the `ddMin` algorithm and compute a minimal failing input from an initial failing input `[G,T,A,C,C,A,G,C]`. Clearly state what happens at *each step* of the algorithm and what the final result is. Correct solutions without explanation will not be given the full score.

Assignment 3 Formal Specification (1) (7p)

Consider the following Dafny program:

```
method AlwaysEven(x : int) returns (y : int)
ensures y%2 == 0;
{
 if (x%2 == 0)
  { y := x; }
else
  { y := (x-1);}
y := 2*y;
}
```

Next, suppose we want to run the following snippet of Dafny code:

```
method Test(){
  var m := AlwaysEven(2);
  var n := AlwaysEven(3);
  assert m == n;
}
```

(a) The above code will cause a Dafny compiler error:

```
Error:  assertion violation
```

Explain why.

(b) Fix `AlwaysEven` so that Dafny would be able to prove the assertion.

Assignment 4 (Formal Specification (2)) (15p)

For this question we consider a simple model of an airline ticket system written in Dafny. The model consists of two classes: `Ticket` and `Flight`:

```
class Ticket{
  var bookingRef : int;
  var checkedIn : bool;

  constructor (ref : int){}
}

class Flight{
  var seats : int;
  var nextBookingRef : int;
  var passports : array<int>;

  constructor(noSeats : int){}

  method IssueTicket(passportNo : int) returns (t : Ticket){}

  method CheckIn(passportNo : int, ticket : Ticket)
  modifies ticket;
  . . .
  {
    ticket.checkedIn := true;
  }
}
```

The `Ticket` class simply models a ticket, with a booking reference number and a boolean field stating if the traveller has checked in. The `Flight` class keeps track of how many seats there are on the flight (the `seat` field) and how many bookings has been made (the `nextBookingRef` field). Tickets issued for a given flight have `bookingRef`s in sequence, starting from 0, up to `seats-1`. The array `passports` maps the booking reference for each ticket issued so far to a corresponding passport number.

(a) Complete the body and specification of the constructor method for the `Ticket` class. A newly issued ticked should of course not yet be checked in.

(b) Complete the body and specification for the constructor method of the `Flight` class. The array `passports` should be initialised to 0 everywhere.
Hint: You might want to provide a predicate capturing the allowed values of the fields for `Flight` objects.

Continued on next page!

(c) Write down a specification and body for the `IssueTicket` method in the `Flight` class. The `Ticket` returned should have an appropriate `bookingRef` and the passport number should be appropriately connected to the ticket with that booking reference. Passport numbers have to be four digit positive numbers. Furthermore, at most one ticket can be purchased by each individual passenger.

(d) Provide a specification for the `CheckIn` method in the `Flight` class. The specification must be sufficiently strict so that a traveller is only allowed to check-in if providing a ticket and passport where the booking reference is valid for this flight, and the passport number provided at check-in matches the passport number associated with the ticket.

Assignment 5 (Formal Verification) (14p)

The following small Dafny program computes the k^{th} positive even number, where the *first* even number is defined to be 0, the *second* is defined to be 2, the *third* is 4 and so on.

```
method kthEven(k : int) returns (e : int)
requires ?
ensures ?
 {
   e := 0;
   var i := 1;
   while (i < k)
   invariant ?
   {
     e := e + 2;
     i := i + 1;
   }
 }
```

(a) Complete the specification of the program by providing suitable `requires` and `ensures` clauses.

(b) State a suitable loop invariant for the program.

(c) State a suitable variant for the loop.

(d) Prove that the program satisfies the specification using the weakest precondition calculus.

(total 60p)