Examples

Testing

Decision coverage

Write down two test-cases for the program below. Your test cases should satisfy decision coverage.

```
int method1(int x, int y)
{
        int res = 0;
        if((x == 0) || (x > y))
                res = y;
        if (isEven(x))
                res = x/2;
        return res;
}
```

Decision coverage (Solution)

Write down two test-cases for the program below. Your test cases should satisfy decision coverage.

```
int method1(int x, int y)
{
        int res = 0;
        if((x == 0) || (x > y))
                res = y;
        if (isEven(x))
                res = x/2;
        return res;
}
```

Solution: Each decision in the program needs at least one test case where it evaluates to **true** and one where it evaluates to **false**

- {x --> 3, y --> 0} (First decision is true, the second is false)
- {x --> 4, y --> 15} (First decision is false, the second is true)

MCDC Criteria

Consider the following piece of (Java) code.

Construct a set of test cases which satisfies MCDC criteria.

```
int method2(int a, int b, int c)
{
        if ( (a < 3) || (b > c && c == 5) )
                return a;
        else
                return c;
}
```

MCDC Criteria (Solution)

Consider the following piece of (Java) code.

Construct a set of test cases which satisfies MCDC criteria.

```
int method2(int a, int b, int c)
{
        if ( (a < 3) || (b > c && c == 5) )
                return a;
        else
                return c;
}
```

Solution

{a = 4, b = 1, c = 5}

{a = 1, b = 1, c = 5}

{a = 4, b = 7, c = 5}

{a =4, b = 7, c = 2}

Minimization using DDMin

Consider a method that takes an array of integers as input, and computes a code that it returns as a result. The method fails if the input array consists of two identical even numbers.

For example, the method fails when the input array is [1, 2, 8, 6, 6, 2, 8, 5],

[2, 6, 7, 7, 5, 2].

Simulate a run of the ddMin algorithm and compute a minimal failing input from the following initial failing input: [1,2,8,6,6,2,8,5].

Minimization using DDMin (Solution)

(b) Start with granularity $n = 2$ and sequence `[1,2,8,6,6,2,8,5]`.

The number of chunks is 2
==> $\texttt{n}: 2, [1, 2, 8, 6]$ PASS (take away second chunk)
==> $\texttt{n}: 2, [6, 2, 8, 5]$ PASS (take away first chunk)

Increase number of chunks to $min(n * 2,\ len([1, 2, 8, 6, 6, 2, 8, 5])) = 4$
==> $\texttt{n}: 4, [8, 6, 6, 2, 8, 5]$ FAIL (take away first chunk)

Adjust number of chunks to $max(n - 1, 2) = 3$
==> $\texttt{n}: 3, [6, 2, 8, 5]$ PASS (take away first chunk)
==> $\texttt{n}: 3, [8, 6, 8, 5]$ FAIL (take away second chunk)

Adjust number of chunks to $max(n - 1, 2) = 2$
==> $\texttt{n}: 2, [8, 5]$ PASS (take away first chunk)
==> $\texttt{n}: 2, [8, 6]$ PASS (take away second chunk)

Increase number of chunks to $min(n * 2,\ len([8, 6, 8, 5])) = 4$
==> $\texttt{n}: 4, [6, 8, 5]$ PASS (take away first chunk)
==> $\texttt{n}: 4, [8, 8, 5]$ FAIL (take away second chunk)

Minimization using DDMin (Solution

Increase number of chunks to $min(n * 2,\ len([8, 6, 8, 5]) = 4$
$==>$ n : 4, [6, 8, 5] PASS (take away first chunk)
$==>$ n : 4, [8, 8, 5] FAIL (take away second chunk)

Adjust number of chunks to $max(n - 1, 2) = 3$
$==>$ n : 3, [8, 5] PASS (take away first chunk)
$==>$ n : 3, [8, 5] PASS (take away second chunk)
$==>$ n : 3, [8, 8] FAIL (take away third chunk)

Adjust number of chunks to $max(n - 1, 2) = 2$

$==>$ n : 2, [8] PASS (take away first chunk)
$==>$ n : 2, [8] PASS (take away second chunk)

As $n == len([8, 8])$ the algorithm terminates with 1-minimal failing input [8, 8]

Formal Specification: Logic

Consider the following propositional logic formula, where p and q are Boolean variables. Is the formula satisfiable? Is the formula valid? Show and explain why?

$$\mathbf{(p \land q) \land (\neg p \lor q)}$$

Formal Specification: Logic (Solution)

Consider the following propositional logic formula, where p and q are Boolean variables. Is the formula satisfiable? Is the formula valid? Show and explain why?

$$(p \land q) \land (\neg p \lor q)$$

Solution

p	q	$\neg p$	$p \land q$	$\neg p \lor q$	$(p \land q) \land (\neg p \lor q)$
T	T	F	T	T	T
T	F	F	F	F	F
F	T	T	F	T	F
F	F	T	F	T	F

Formal Specification

Define the **pre** and **post** conditions for the following linearSearch method formally.

```
method linearSearch( a : array<int>, element : int)
    returns (index : int)
       {
             .....
             .....
       }
```

Informal description: the linearSearch method should take a sorted array and search for the given number in the array. It should **return −1** if the given number is not present in the array, and otherwise **return an index** such that the number is at that place in the array.