Loop Invariants

Simple loop property

```
def foo(a: Array[Int]): Array[Int] = {
    require(a.length > 0)
    val a2 = Array.fill(a.length)(0)
    var i = 0
    (while(i < a.length) {
        a2(i) = a(i)
        i = i + 1
    }) invariant (???)
    a2
} ensuring (res => res(0) == a(0))
```

Simple loop property: ensuring the post

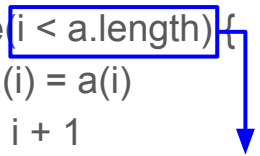
```
def foo(a: Array[Int]): Array[Int] = {
    require(a.length > 0)
    val a2 = Array.fill(a.length)(0)
    var i = 0
    (while(i < a.length) {
        a2(i) = a(i)
        i = i + 1
    }) invariant ((i > 0) ==> (a2(0) == a(0)))
    a2
} ensuring (res => res(0) == a(0))
```

Simple loop property: array usages

```
def foo(a: Array[Int]): Array[Int] = {
    require(a.length > 0)
    val a2 = Array.fill(a.length)(0)
    var i = 0
    (while(i < a.length) {
        a2(i) = a(i)
        i = i + 1
    }) invariant (i >= 0 && (i > 0) ==> (a2(0) == a(0)))
    a2
} ensuring (res => res(0) == a(0))
```

Simple loop property: array usages

```
def foo(a: Array[Int]): Array[Int] = {
  require(a.length > 0)
  val a2 = Array.fill(a.length)(0)
  var i = 0
  (while(i < a.length) {
    a2(i) = a(i)
    i = i + 1
  }) invariant (i < a.length && i >= 0 && (i > 0) ==> (a2(0) == a(0)))
  a2
} ensuring (res => res(0) == a(0))
```



Simple loop property: array usages

```
def foo(a: Array[Int]): Array[Int] = {
    require(a.length > 0)
    val a2 = Array.fill(a.length)(0)
    var i = 0
    (while(i < a.length) {
        a2(i) = a(i)
        i = i + 1
    }) invariant (a2.length == a.length && i >= 0 && (i > 0) ==> (a2(0) == a(0)))
    a2
} ensuring (res => res(0) == a(0))
```

Complex loop

```
def binarySearch(a: Array[BigInt], key: BigInt): Int = ({
  require(a.length > 0 && forall((i: Int, j: Int) => (i >= 0 && j >= 0 && i < a.length && j < a.length && i < j) ==> (a(i) <= a(j))))
  var low = 0
  var high = a.length - 1
  var res = -1
  (while (low <= high && res == -1) {
    val o = if ((high & 1) == 1 && (low & 1) == 1) 1 else 0
    val i = high / 2 + low / 2 + o

    if (a(i) == key) res = i
    else if (a(i) > key) high = i - 1
    else low = i + 1
  }) invariant(???)
  res
}) ensuring(res => (res == -1 && forall((i: Int) => (0 <= i && i < a.length) ==> (a(i) != key))) || (res != -1 && a(res) == key))
```

Complex loop

```
def binarySearch(a: Array[BigInt], key: BigInt): Int = ({
  require(a.length > 0 && forall((i: Int, j: Int) => (i >= 0 && j >= 0 && i < a.length && j < a.length && i < j) ==> (a(i) <= a(j))))
    ...
  (while (low <= high && res == -1) {
    ...
  }) invariant(???)
  res
}) ensuring(res => (res == -1 && forall((i: Int) => (0 <= i && i < a.length) ==> (a(i) != key))) || (res != -1 && a(res) == key))
```

Complex loop: check post structure

```
def binarySearch(a: Array[BigInt], key: BigInt): Int = ({
  require(a.length > 0 && forall((i: Int, j: Int) => (i >= 0 && j >= 0 && i < a.length && j < a.length && i < j) ==> (a(i) <= a(j))))
    ...
  (while (low <= high && res == -1) {
    ...
  }) invariant(
    (res == -1 && ???) ||
    (res != -1 && ???)
  )
  res
}) ensuring(res => (res == -1 && forall((i: Int) => (0 <= i && i < a.length) ==> (a(i) != key))) || (res != -1 && a(res) == key))
```

Complex loop: ensuring the post

```
def binarySearch(a: Array[BigInt], key: BigInt): Int = ({
  require(a.length > 0 && forall((i: Int, j: Int) => (i >= 0 && j >= 0 && i < a.length && j < a.length && i < j) ==> (a(i) <= a(j))))
    ...
  (while (low <= high && res == -1) {
    ...
  }) invariant(
    (res == -1 && ???) ||
    (res != -1 && a(res) == key)
  )
  res
}) ensuring(res => (res == -1 && forall((i: Int) => (0 <= i && i < a.length) ==> (a(i) != key))) || (res != -1 && a(res) == key))
```

Complex loop: post and invariant

```
def binarySearch(a: Array[BigInt], key: BigInt): Int = ({
  require(a.length > 0 && forall((i: Int, j: Int) => (i >= 0 && j >= 0 && i < a.length && j < a.length && i < j) ==> (a(i) <= a(j))))
    ...
  (while (low <= high && res == -1) {
    ...
  }) invariant(
    (if (res == -1)
      forall((i: Int) => (0 <= i && i < low) ==> (a(i) != key)) &&
      forall((i: Int) => (high + 1 <= i && i < a.length) ==> (a(i) != key))
    else
      res >= 0 && res < a.length && a(res) == key)
    )
  res
}) ensuring(res => (res == -1 && forall((i: Int) => (0 <= i && i < a.length) ==> (a(i) != key))) || (res != -1 && a(res) == key))
```

Complex loop: array usages

```
def binarySearch(a: Array[BigInt], key: BigInt): Int = ({
  require(a.length > 0 && forall((i: Int, j: Int) => (i >= 0 && j >= 0 && i < a.length && j < a.length && i < j) ==> (a(i) <= a(j))))
    ...
  (while (low <= high && res == -1) {
    ...
  }) invariant(
    0 <= low && low <= high + 1 && high < a.length && (
    (if (res == -1)
      forall((i: Int) => (0 <= i && i < low) ==> (a(i) != key)) &&
      forall((i: Int) => (high + 1 <= i && i < a.length) ==> (a(i) != key))
    else
      res >= 0 && res < a.length && a(res) == key)
    )
  res
}) ensuring(res => (res == -1 && forall((i: Int) => (0 <= i && i < a.length) ==> (a(i) != key))) || (res != -1 && a(res) == key))
```