using System;

//using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Diagnostics;

using System.Diagnostics.CodeAnalysis;

using System.Runtime.CompilerServices;

using System.Collections.Generic;

using DataStructures.Utility;

using PexAPIWrapper;

namespace DataStructures

{

public class StackAbstract

{

int \_top;

int \_count = 0;

bool \_containsX = false;

public StackAbstract()

{

\_count = 0;

}

public int Count

{

get {

Int oldCount = \_count;

Bool oldContains = \_containsX;

Int oldTop = \_top;

Int ret = \*;

PexAssert.IsTrue(true) ; PexAssume.IsTrue( oldCount == \_count && oldContains == \_contains && oldTop == \_top && ret == \_count );

\_count = Pex

\_top =

ret = PexChoose.Value<int>("\_ret");

return ret;

}

// return \_count;

}

public int Top

{

get { return \_top; }

}

public void Push(int x)

{

Int oldX = x;

Int oldCount = \_count;

Bool oldContainsX = \_containsX ;

Int oldTop = \_top;

PexAssert.IsTrue(true);

PexAssume.IsTrue(( \_containsX == true && (\_top == oldX) && (\_count == 1 + oldCount) && (oldX == x));

\_containsX = PexChoose<bool>.Value(“\_containsX”);

\_top = PexChoose<int>.Value(“\_top”);

\_count

}

public int Pop()

{

Assume(ret ..)

Ret = pexChoose

}

}

}

PostCondition for push:

((New\_s1ContainsX == true) && (New\_Top == Old\_x) && (New\_s1Count == 1 + Old\_s1Count) && (New\_Top == New\_x))

Precondition for pop:

Count > 0

PostCondition for pop:

(New\_s1Count == Old\_s1Count - 1) && ret == top)

using System;

using System.Text;

using Microsoft.Pex.Framework;

using Microsoft.Pex.Framework.Settings;

using Microsoft.Pex.Framework.Exceptions;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using DataStructures;

using DataStructures.Utility;

using PexAPIWrapper;

namespace DataStructures

{

public class StackClient

{

StackAbstract <int> stack = new StackAbstract <int>();

Public StackClient(StackAbstract stack )

{  
 this.stack = stack  
 }

public string ToBinaryStack(int n)

{

Stack<int>stack = new Stack<int>();

string output = "";

while (n > 0)

{

stack.Push(n % 2);

n /= 2;

}

while (stack.Count > 0)

{

output += stack.Pop().ToString();

}

return output;

}

public string ToBinaryStackSummary(int n)

{

string output = "";

int oldBit = -1;

int currBit = -1;

int oldCount = 0;

while (n > 0)

{

currBit = n % 2;

stack.Push(currBit);

n /= 2;

}

while (this.stack.Count > 0)

{

Int top = this.stack.Pop()

output += top.ToString();

}

return output;

}

}

}

N results

0 “”

1 “1”

2 “10”

5 “101”

21 “10101”

340 “101010100”