TÖL212M Lokapróf

Nafn/Name: ……..

Háskólatölvupóstfang/University Email: ……..

1. Engin hjálpargögn eru leyfileg.
2. Skrifið svörin á þessar síður, ekki á önnur blöð og ekki á baksíður.
3. Ef svarið kemst ekki fyrir á tilteknu svæði má skrifa á auðar síður aftast, en þá skalt þú láta vita af því með því að skrifa tilvísun í tiltekið svæði, til dæmis “framhald á blaðsíðu 14”.
4. Forðist að skemma eða rífa þessar síður, þær þurfa að fara gegnum skanna. Skrifið skýrt með **dökku letri** og ekki skrifa í spássíur.
5. Baksíður **verða ekki skannaðar** og má nota fyrir krass. **Ekki verður tekið tillit til** svara sem skrifuð eru á baksíður.
6. Prófið skiptist í **hluta**. Svarið **8** spurningum í heild og að minnsta kosti **tilteknum lágmarksfjölda** í hverjum hluta.
7. Ef þú svarar fleiri en **8** spurningum þá verður einkunn þín reiknuð sem **meðaltal allra svara** nema þú **krossir skýrt út** svör sem þú vilt ekki að gildi. Þú verður að krossa út allt svarið, ekki aðeins hluta þess.
8. Munið að öll Dafny föll þurfa **notkunarlýsingu** með **requires/ensures**. Allar lykkjur þurfa **invariant** sem dugar til að rökstyðja.
9. Munið að öll Java föll þurfa **notkunarlýsingu** með Notkun/Fyrir/Eftir. Allar lykkjur þurfa **fastayrðingu** sem dugar til að rökstyðja.
10. Munið að nota viðeigandi **innfellingu** í öllum forritstexta.

Hluti I – Helmingunarleit o.fl.

Svarið að minnsta kosti tveimur spurningum í þessum hluta – Munið að svara a.m.k. 8 spurningum í heild

1.

Skrifið fall í Dafny sem leitar með helmingunarleit í heiltalnafylki, a, sem raðað er í minnkandi röð, að fremsta sæti sem inniheldur neikvæða tölu, þ.e. tölu minni en núll. Ef ekkert slíkt sæti er til skal skila -1.

Svar:

2.

Skrifið endurkvæmt helmingunarleitarfall í Dafny sem leitar í svæði í heiltalnafylki sem raðað er í vaxandi röð að aftasta sæti innan svæðisins sem inniheldur neikvæða tölu. Ef ekkert slíkt sæti er til innan svæðisins skal skila -1.

Svar:

3.

Skrifið endurkvæmt helmingunarleitarfall í Java sem hefur eftirfarandi lýsingu:

```
// Notkun: int k = find(a,i,n,x);
// Fyrir:  0 <= i <= i+n <= a.length, x er heiltala,
//         a[i..i+n) er í minnkandi röð. (Í Dafny myndum við
//         skrifa a[i..i+n] í stað a[i..i+n).)
// Eftir:  i <= k <= i+n.
//         a[i..k) >= x > a[k..i+n).
//         Fylkið a er óbreytt.
static int find( int[] a, int i, int n, int x )
{
  …
}
```

Svar:

Hluti II – Quicksort o.fl.

Svarið að minnsta kosti einni spurningu í þessum hluta – Munið að svara a.m.k. 8 spurningum í heild

4.

Gerið ráð fyrir að til sé Dafny fall með eftirfarandi lýsingu:

```
method QuickSelect( a: array<int>, i: int, k: int, j: int )
   modifies a;
   requires 0 <= i <= k < j <= a.Length;
   ensures forall r | i <= r < k :: a[r] <= a[k];
   ensures forall r | k < r < j :: a[r] >= a[k];
   ensures multiset(a[i..j]) == old(multiset(a[i..j]));
   ensures a[..i] == old(a[..i]);
   ensures a[j..] == old(a[j..]);
```

Skrifið Quicksort fall sem notar þetta fall sem hjálparfall. Ekki forrita QuickSelect fallið hér.

Svar:

5.

Forritið fallið QuickSelect sem lýst er að ofan. Munið að allar lykkjur þurfa invariant. Nota má lykkju eða endurkvæmni eða hvort tveggja.

Svar:

6.

Gerið ráð fyrir að til sé Dafny fall með eftirfarandi lýsingu:

```
method TriPartition( a: array<int>, i: int, j: int )
      returns( p: int, q: int )
   modifies a;
   requires 0 <= i < j <= a.Length;
   requires j-i > 1;
   ensures i <= p < q < j;
   ensures a[p] <= a[q];
   ensures forall r | i <= r < p :: a[r] <= a[p];
   ensures forall r | p < r < q :: a[p] <= a[r] <= a[q];
   ensures forall r | q < r < j :: a[q] <= a[r];
   ensures multiset(a[i..j]) == old(multiset(a[i..j]));
   ensures a[..i] == old(a[..i]);
   ensures a[j..] == old(a[j..]);
```

Skrifið Quicksort fall sem notar þetta fall sem hjálparfall. Ekki forrita TriPartition fallið.

Svar:

Hluti III – Tvíleitartré

Svarið að minnsta kosti tveimur spurningum í þessum hluta – Munið að svara a.m.k. 8 spurningum í heild
Aftast í prófinu eru lýsingar á helstu föllum í skránni BST.dfy.

7.

Gerið ráð fyrir skilgreiningunni

```
datatype BST = BSTEmpty | BSTNode(BST,int,BST)
```

eins og í skránni okkar BST.dfy. Gerið einnig ráð fyrir föllunum IsTreePath, TreeSeq, PreSeq, MidSeq, PostSeq, PreSeqIncluding, o.s.frv.
Skrifið fall sem leitar í tvíleitartré og skilar tilvísun á fremsta hnút í milliröð sem inniheldur jákvæða tölu (tölu stærri en núll), eða skilar BSTEmpty ef slíkur hnútur finnst ekki í leitartrénu.
Munið að skrifa fulla lýsingu með requires/ensures og skrifa invariant fyrir lykkjuna ef þið notið lykkju.

Svar:

8. **Leysið sama vandamál og að ofan, en núna í Java. Notið gamalkunna skilgreiningu á trjáhnútum, þ.e.:**

```
public class BSTNode {
  private BSTNode left,right;
  private int val;
  public BSTNode( BST a, int x, BST b )
  { left=a; val=x; right=b; }
  public static left( BSTNode t ) { return t.left; }
  public static right( BSTNode t ) { return t.right; }
  public static rootValue( BSTNode t ) { return t.val; }
}
```

(Ég sleppi hér Notkun/Fyrir/Eftir til að spara pláss því þau eru gamalkunnug og augljós. Athugið að það þýðir ekki að nemendur megi sleppa að skrifa gamalkunnugar og augljósar lýsingar fyrir sína klasa.)

Svar:

9. **Skrifið fall í Dafny sem tekur þrjú viðföng, tvíleitartré t og heiltölur a og b með a<b og skilar true ef til er tala x í trénu þannig að a<x<b en skilar false annars. Þið megið nota lykkju eða endurkvæmni, að því tilskildu að rökstuðningur sé réttur (þ.e. rétt requires, ensures og invariant).**

   Svar:

10.

Íhugið eftirfarandi Dafny forritstexta.

```
include "BST.dfy"

predicate IsReverse( x: seq<int>, y: seq<int> )
{
    |x| == |y| &&
    forall i | 0 <= i < |x| :: x[i] == y[|x|-i-1]
}

method RevSeq( t: BST ) returns( r: seq<int> )
    ensures IsReverse(r,TreeSeq(t));
{
    …
}
```

Forritið stofn fallsins RevSeq.

Svar:

Hluti IV – Ýmislegt

Svarið að minnsta kosti einni spurningu í þessum hluta – Munið að svara a.m.k. 8 spurningum í heild

11.

Forritið stofninn á fallinu að neðan. Ekki nota lykkju heldur endurkvæmni.

```
method IsSorted ( a: array<int>, i: int, j: int )
    returns( r: bool )
  decreases j-i;
  requires 0 <= i <= j <= a.Length;
  ensures r <==>
     forall p,q | i <= p < q < j :: a[p] <= a[q];
```

Svar:

12.

Forritið stofninn á fallinu að neðan. Notið lykkju en ekki endurkvæmni.

```
method IsSorted ( a: array<int>, i: int, j: int )
    returns( r: bool )
  requires 0 <= i <= j <= a.Length;
  ensures r <==>
     forall p,q | i <= p < q < j :: a[p] <= a[q];
```

Svar:

(auð blaðsíða/empty page)

(auð blaðsíða/empty page)

Mikilvæg föll og tög í BST.dfy

```
// Skilgreining BST:
datatype BST = BSTEmpty | BSTNode(BST,int,BST)
// Gildi af tagi BST eru tvíundartré.

// Skilgreining trjáslóða:
newtype dir = x | 0 <= x <= 1
// Trjáslóðir eru af tagi seq<dir>.

// Öll þau fyrirbæri sem hér er lýst má nota í
// röksemdafærslu, þ.e. í requires/ensures/invariant.
// Þau föll sem hafa Notkun/Fyrir/Eftir má nota í
// raunverulegum útreikningum en hin, sem hafa
// Notkun/Fyrir/Gildi, eru einungis nothæf í
// röksemdafærslu.

// Notkun: var t := BSTEmpty;
// Fyrir:  Ekkert.
// Eftir:  t er tómt tvíundartré.

// Notkun: var t := BSTNode(p,x,q);
// Fyrir:  p og q eru tvíundartré.
// Eftir:  t er tvíundartré með x í rót,
//         með p sem vinstra undirtré og
//         með q sem hægra undirtré.

// Notkun: var x = RootValue(t);
// Fyrir:  t er tvíundartré, ekki tómt.
// Eftir:  x er gildið í rót t.

// Notkun: var l = Left(t);
// Fyrir:  t er tvíundartré, ekki tómt.
// Eftir:  l er vinstra undirtré t.

// Notkun: var r = Right(t);
// Fyrir:  t er tvíundartré, ekki tómt.
// Eftir:  r er hægra undirtré t.
```

```
// Notkun: TreeIsSorted(t)
// Fyrir:  t er tvíundartré.
// Gildi:  satt ef t er tvíleitartré, þ.e. í vaxandi
//         millirröð, ósatt annars.

// Notkun: TreeSeq(t)
// Fyrir:  t er tvíundartré.
// Gildi:  Runa gildanna í t í millirröð,
//         af tagi seq<int>.

// Notkun: IsTreePath(t,p)
// Fyrir:  t er tvíundartré, p er trjáslóð.
// Gildi:  Satt ef p er slóð innan t, annars ósatt.
// Ath.:   Trjáslóðin [] er slóð innan allra trjáa.
//         Ef p==[0]+q þá er p trjáslóð innan t ef t
//         er ekki tómt og q er trjáslóð innan Left(t).
//         Ef p==[1]+q þá er p trjáslóð innan t ef t
//         er ekki tómt og q er trjáslóð innan Right(t).

// Notkun: Subtree(t,p)
// Fyrir:  t er tvíundartré, p er trjáslóð innan t.
// Gildi:  Undirtréð innan t sem p vísar á.
// Ath.:   Ef p er [] þá er skilagildið t, ef p==[0]+q
//         þá er það Subtree(Left(t),q) og ef p==[1]+q
//         þá er það Subtree(Right(t),q).

// Notkun: PreSeq(t,p)
// Fyrir:  t er tvíundartré, p er trjáslóð innan t.
// Gildi:  Runa þeirra gilda í millirröð  innan t sem
//         eru fyrir framan undirtréð sem p vísar á.

// Notkun: PostSeq(t,p)
// Fyrir:  t er tvíundartré, p er trjáslóð innan t.
// Gildi:  Runa þeirra gilda í millirröð  innan t sem
//         eru fyrir aftan undirtréð sem p vísar á.

// Notkun: MidSeq(t,p)
// Fyrir:  t er tvíundartré, p er trjáslóð innan t.
// Gildi:  Runa gildanna í millirröð sem eru í
//         undirtrénu sem p vísar á. Sama og
//         TreeSeq(Subtree(t,p))

// Fyrir sérhverja trjáslóð p innan t gildir:
//   TreeSeq(t) == PreSeq(t,p)+MidSeq(t,p)+PostSeq(t,p)
```

```
// Notkun: PreSeqIncluding(t,p)
// Fyrir:  t er tvíundartré og p er trjáslóð innan t
//         sem vísar á ekki-tómt undirtré.
// Gildi:  Runa þeirra gilda í t, í milliröð, sem eru
//         í hnútunum fram til hnútsins sem p vísar á
//         að þeim hnút meðtöldum.

// Notkun: PostSeqExcluding(t,p)
// Fyrir:  t er tvíundartré og p er trjáslóð innan t
//         sem vísar á ekki-tómt undirtré.
// Gildi:  Runa þeirra gilda í t, í milliröð, sem eru
//         í hnútunum fyrir aftan hnútinn sem p vísar
//         á.

// Notkun: PreSeqExcluding(t,p)
// Fyrir:  t er tvíundartré og p er trjáslóð innan t
//         sem vísar á ekki-tómt undirtré.
// Gildi:  Runa þeirra gilda í t, í milliröð, sem eru
//         í hnútunum fyrir framan hnútinn sem p
//         vísar á.

// Notkun: PostSeqIncluding(t,p)
// Fyrir:  t er tvíundartré og p er trjáslóð innan t
//         sem vísar á ekki-tómt undirtré.
// Gildi:  Runa þeirra gilda í t, í milliröð, sem eru
//         í hnútnum sem p vísar á eða fyrir aftan
//         hann.

// Fyrir sérhverja trjáslóð p sem vísar á ekki-tómt undirtré
// innan t gildir:
//   TreeSeq(t) == PreSeqExcluding(t,p)+PostSeqIncluding(t,p)
//   TreeSeq(t) == PreSeqIncluding(t,p)+PostSeqExcluding(t,p)
```