TÖL212M Miðmisserispróf
TÖL212M Midterm Exam

Nafn/Name: ……..

Háskólatölvupóstfang/University Email: ……..

1. No help materials are allowed.
   Engin hjálpargögn eru leyfileg.
2. Write your answers on these pages, not in an exam book.
   Skrifið svörin á þessar síður, ekki í prófbók.
3. If the answer does not fit on the allotted space you may write on the empty pages at the end, but in that case you should write an indication to that effect on the space allotted for the answer, for example “continued on page 14”.
   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. Refrain from mutilating or tearing these pages, they have to go through a scanner. Write clearly with **dark letters** and do not write in the margins.
   Forðist að skemma eða rífa þessar síður, þær þurfa að fara gegnum skanna. Skrifið skýrt með **dökku letri** og ekki skrifa í spássíur.
5. The backs of the pages **will not be scanned** and can be used for scratch. Any answers written on the backs **will be ignored**.

Baksíður **verða ekki skannaðar** og má nota fyrir krass. **Ekki verður tekið tillit til** svara sem skrifuð eru á baksíður.

6. Prófið inniheldur fleiri en 5 dæmi en aðeins skal skila 5 þeirra. Sé skilað fleiri en 5 verður einkunnin reiknuð sem meðaltal skilaðra dæma.
   The exam contains more than 5 questions but only 5 need to be answered. If more than 5 are answered then the grade will be computed as the average of the answered questions.
7. Munið að öll Dafny föll þurfa **notkunarlýsingu** með **requires/ensures**. Allar lykkjur þurfa **invariant** sem dugar til að rökstyðja. **decreases** klausur eru valfrjálsar.
   Remember that all Dafny functions and methods need a usage description with **requires/ensures.** All loops need an **invariant** that is sufficient to validate. **decreases** clauses are optional.
8. Remember that all Java functions need a **description** with Usage/Pre/Post or Usage/Pre/Value and all loops need an invariant.
   Munið að öll Java föll þurfa **notkunarlýsingu** með Notkun/Fyrir/Eftir eða Notkun/Fyrir/Gildi og allar lykkjur þurfa fastayrðingu.
9. Remember to use proper **indentation** in all program code.
   Munið að nota viðeigandi **innfellingu** í öllum forritstexta.

1. **Skrifið fall í Dafny sem leitar með helmingunarleit í heiltalnarunu, a, sem raðað er í minnkandi röð að aftasta sæti sem inniheldur gildi ≥ 100. Ef ekkert slíkt sæti er til skal skila -1.**
   Write a method in Dafny that searches with binary search in an integer sequence a, which is in descending order, for the last position that contains a value ≥ 100. If no such position exists then return -1.

2. **Skrifið helmingunarleitarfall í Dafny sem tekur þrjú viðföng, heiltalnarunu s í vaxandi röð, tvær heiltölur a og b með a<=b, sem skilar vísi á eitthvert sæti í rununni sem inniheldur gildi x þannig að a<=x<=b. Ef ekkert slíkt sæti er til skal skila -1.**
   Write a binary search method in Dafny that takes three arguments, an integer sequence s in ascending order, two integers a and b with a<=b, which returns an index to some position in the sequence containing a value x such that a<=x<=b. If no such position exists then return -1.

3. **Reiknið með að til sé eftirfarandi Dafny fall.**
   Assume the existence of the following Dafny method.

```
method Partition( s: seq<int> )
     returns ( a: seq<int>
             , p: int
             , b: seq<int>
             , q: int
             , c: seq<int>
             )
   requires |s| >= 2;
   ensures p <= q;
   ensures forall z | z in a :: z < p;
   ensures forall z | z in b :: p <= z <= q;
   ensures forall z | z in c :: q < z;
   ensures multiset(s) ==
             multiset(a)+
             multiset{p}+
             multiset(b)+
             multiset{q}+
             multiset(c);
```

Skrifið quicksort fall sem tekur runu heiltalna (seq<int>) sem viðfang og skilar vaxandi runu sömu talna og notar þetta Partition fall sem hjálparfall. Ekki þarf að forrita Partition fallið.
Write a quicksort method that takes a sequence of integers (seq<int>) as an argument and returns an ascending sequence of the same numbers and uses this Partition method as a helper method. The Partition method need not be written.

4. **Forritið fallið Partition sem lýst er að ofan. Munið að allar lykkjur þurfa invariant.**
   Program the Partition method described above. Remember that all loops need an invariant.

5. **Gerið ráð fyrir skilgreiningunni**

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

eins og í skránni okkar BST.dfy. Gerið einnig ráð fyrir föllunum IsTreePath, PreSeq, MidSeq, o.s.frv., sem nota trjáslóðir til að búta tré (skilgreiningar eru aftast í prófinu). Skrifið fall sem leitar með lykkju í tvíleitartré og skilar tilvísun á aftasta hnút í milliröð sem inniheldur gildi <100, eða skilar BSTEmpty ef slíkur hnútur finnst ekki í leitartrénu.
Munið að skrifa fulla lýsingu með requires/ensures og skrifa invariant fyrir lykkjuna.
Assume the definition

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

as in our file BST.dfy. Assume also the functions IsTreePath, PreSeq, MidSeq, etc., that use tree paths to segment a tree (definitions are at the end of the exam).
Write a function that searches using a loop in a binary search tree and returns the rightmost node containing a <100, or returns BSTEmpty is no such node exists in the search tree. Remember to write a full description with requires/ensures and write an invariant for the loop.

6. **Gerið ráð fyrir sömu BST skilgreiningum og að ofan. Skrifið Dafny fall sem tekur þrjú viðföng, tvíleitartré t, og heiltölur a, b með a<=b. Fallið skal skila undirtré t sem hefur gildi x í rótinni þannig að a<=x<=b, ef slíkt undirtré er til. Annars skal fallið skila BSTEmpty. Hafa má fleiri skilagildi, ef ástæða þykir til, en það er ekki nauðsynlegt.**
   Assume the same BST definitions as above. Write a Dafny method that takes three arguments, a binary search tree t, and integers a, b with a<=b. The method should return a subtree t that has a root value x such that a<=x<=b, if such a subtree exists. Otherwise return BSTEmpty. You may have more return values if you wish, but that is not necessary.

7. Klárið að forrita eftirfarandi Dafny fall.
Finish programming the following Dafny method.

```
method Min( x: seq<int> ) returns( m: int )
    requires |x| > 0;
    ensures m in x;
    ensures forall j | 0 <= j < |x| :: x[j] >= m;
{
    // Hér vantar stofn fallsins – Missing body









}
```

Mikilvæg föll og tög í BST.dfy
Important functions and types in BST.dfy

```
// Skilgreining BST / Definition of BST:
datatype BST = BSTEmpty | BSTNode(BST,int,BST)
// Gildi af tagi BST eru tvíundartré.
// Values of type BST are binary trees.

// Skilgreining trjáslóða / The definition of tree paths:
newtype dir = x | 0 <= x <= 1
// Trjáslóðir eru af tagi seq<dir>.
// Tree paths are of type 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.

// All the items described here can be used in reasoning,
// i.e. in requires/ensures/invariant.
// Those functions that have Usage/Pre/Post can be used in
// real computations, but the others, that have
// Usage/Pre/Value, can only be used in reasoning.

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

// Usage:  var t := BSTEmpty;
// Pre:    Nothing.
// Post:   t is an empty binary tree.

// 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.

// Usage:  var t := BSTNode(p,x,q);
// Pre:    p and q are binary trees.
// Post:   t is a binary tree with x in the root,
//         with p as the left subree and
//         with q as the right subtree.
```

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

// Usage:  var x = RootValue(t);
// Pre:    t is a binary tree, not empty.
// Post:   x is the value in the root of t.

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

// Usage:  var l = Left(t);
// Pre:    t is a binary tree, not empty.
// Post:   l is the left subtree of t.

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

// Usage:  var r = Right(t);
// Pre:    t is a binary tree, not empty.
// Post:   r is the right subtree of t.

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

// Usage:  TreeIsSorted(t)
// Pre:    t is a binary tree.
// Value:  true is t is a binary search tree, i.e. in
//         ascending order when traversed inorder,
//         otherwise false.

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

// Usage:  TreeSeq(t)
// Pre:    t is a binary tree.
// Value:  The sequence of the values in t when t is
//         traversed inorder, of type 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).

// Usage:  IsTreePath(t,p)
// Pre:    t is a binary tree, p is a tree path.
// Value:  true is p is a path within t, otherwise false.
// Note:   The path [] is a path within all trees.
//         If p==[0]+q then p is a path within t if t is not
//         empty and q is a path within Left(t).
//         If p==[1]+q then p is a path within t if t is not
//         empty and q is a path within 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).

// Usage:  Subtree(t,p)
// Pre:    t is a binary tree, p is a tree path.
// Value:  The subtree within t that p refers to.
// Note:   If p is [] then the return value is t, if p==[0]+q
//         then it is Subtree(Left(t),q) and if p==[1]+q
//         then it is Subtree(Right(t),q).

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

// Usage:  PreSeq(t,p)
// Pre:    t is a binary tree, p is a tree path within t.
// Value:  The sequence of the values when traversed inorder
//         that are to the left of the subtree that p refers
//         to.
```

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

// Usage:  PostSeq(t,p)
// Pre:    t is a binary tree, p is a tree path within t.
// Value:  The sequence of the values when traversed inorder
//         that are to the right of the subtree that p refers
//         to.

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

// Usage:  MidSeq(t,p)
// Pre:    t is a binary tree, p is a tree path within t.
// Value:  The sequence of the values inorder that are in
//         the subtree that p refers to. Same as
//         TreeSeq(Subtree(t,p))

// Fyrir sérhverja trjáslóð p innan t gildir:
// For each tree path p within t we have:
//   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.

// Usage:  PreSeqIncluding(t,p)
// Pre:    t is a binary tree and p is a tree path within t
//         that refers to a non-empty subtree.
// Value:  The sequence of values in t, when traversed inorder
//         that are in nodes to the left of the node that p
//         refers to and also including that node.

// 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.
```

```
// Usage:  PostSeqExcluding(t,p)
// Pre:    t is a binary tree and p is a tree path within t
//         that refers to a non-empty subtree.
// Value:  The sequence of values in t, when traversed inorder
//         that are in nodes to the right of the node that p
//         refers to and excluding that node.

// 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.

// Usage:  PreSeqExcluding(t,p)
// Pre:    t is a binary tree and p is a tree path within t
//         that refers to a non-empty subtree.
// Value:  The sequence of values in t, when traversed inorder
//         that are in nodes to the left of the node that p
//         refers to and excluding that node.

// 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.

// Usage:  PostSeqIncluding(t,p)
// Pre:    t is a binary tree and p is a tree path within t
//         that refers to a non-empty subtree.
// Value:  The sequence of values in t, when traversed inorder
//         that are in nodes to the right of the node that p
//         refers to and also including that node.


// Fyrir sérhverja trjáslóð p sem vísar á ekki-tómt undirtré
// innan t gildir:
// For each tree path p that refers to a non-empty subtree
// within t, we have:
//
// TreeSeq(t) == PreSeqExcluding(t,p)+PostSeqIncluding(t,p)
// TreeSeq(t) == PreSeqIncluding(t,p)+PostSeqExcluding(t,p)
// PreSeqExcluding(t,p) == PreSeq(t,p)+TreeSeq(Left(Subtree(t,p)))
// PreSeqIncluding(t,p) == PreSeqExcluding(t,p)+[RootValue(Subtree(t,p))]
// PostSeqExcluding(t,p) == TreeSeq(Right(Subtree(t,p)))+PostSeq(t,p)
// PostSeqIncluding(t,p) == [RootValue(Subtree(t,p))]+PostSeqExcluding(t,p)
```