TÖL212M Lokapróf
TÖL212M Final Exam

Nafn/Name: ……..

Háskólatölvupóstfang/University Email: ……..

1. Engin hjálpargögn eru leyfileg.
   No help materials are allowed.
2. Skrifið svörin á þessar síður, ekki á önnur blöð og ekki á baksíður.
   Write the answers on these pages, not on other pages and not on the back sides.
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”.
   If the answer does not fit on the allocated space you may write on empty pages at the back, but then you should indicate this by referring to the relevant space, for example writing “continued on page 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.
   Refrain from damaging or tearing these pages, they need to be passed through a scanner. Write clearly with **dark letters** and do not write on margins.
5. Baksíður **verða ekki skannaðar** og má nota fyrir krass. **Ekki verður tekið tillit til** svara sem skrifuð eru á baksíður.
   The backs of pages **will not be scanned** and can be used for scratch. Answers written on back pages **will be ignored**.

6. Prófið skiptist í **hluta**. Svarið **8** spurningum í heild og að minnsta kosti **tilteknum lágmarksfjölda** í hverjum hluta.
   The exam is divided into **parts**. Answer **8** questions in total and at least the **required minimum** in each part.
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.
   If you answer more than **8** questions then your grade will be computed as the **average of all answers** unless you **clearly cross out** answers you do not want to count. You must cross out the whole answer, not just part of it.
8. Munið að öll Dafny föll þurfa **notkunarlýsingu** með **requires/ensures**. Allar lykkjur þurfa **invariant** sem dugar til að rökstyðja.
   Remember that all Dafny functions need a **description** with **requires/ensures**. All loops need an **invariant** that is sufficient for supporting its correctness.
9. Munið að öll Java föll þurfa **notkunarlýsingu** með Notkun/Fyrir/Eftir. Allar lykkjur þurfa **fastayrðingu** sem dugar til að rökstyðja.
   Remember that all Java functions need a **description** with Usage/Pre/Post. All loops need an invariant that is sufficient to support its correctness.
10. Munið að nota viðeigandi **innfellingu** í öllum forritstexta.
    Remember to use the appropriate **indentation** in all code.
11. Ekki þarf að kalla á neinar hjálparsetningar í Dafny, jafnvel þótt vera megi að slíkt sé nauðsynlegt svo Dafny samþykki lausnins. Sama gildir um assert.
    You do not need to call any lemmas in Dafny even though that might be necessary for Dafnyt to accept your solution. The same applies to asserts.

Hluti I – Helmingunarleit o.fl.
Part I – Binary Search etc.

Svarið að minnsta kosti tveimur spurningum í þessum hluta – Munið að svara a.m.k. 8 spurningum í heild
Answer at least two questions in this part – Remember to answer at least 8 questions in total

1.

Skrifið fall í Dafny sem leitar með helmingunarleit í heiltalnafylki, a, sem raðað er í vaxandi röð, að fremsta sæti sem inniheldur núll. Ef ekkert slíkt sæti er til skal skila -1.

Write a function in Dafny that uses binary search to find the leftmost position in an ascending integer array a that contains a zero. If no such position exists then return -1.

Svar/Answer:

2.

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

Write a recursive binary search method in Dafny that searches in a section of an integer array that is in descending order for the rightmost position that contains a positive number. If no such position exists return -1.

Svar/Answer:

3.

Skrifið endurkvæmt helmingunarleitarfall í Java sem hefur eftirfarandi lýsingu – Write a recursive binary search function in Java with the following description:

```
// Notkun: int k = find(a,i,n,x);
// Fyrir:  0 <= i <= i+n <= a.length, x er heiltala,
//         a[i..i+n) er í vaxandi 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.

// Usage:  int k = find(a,i,n,x);
// Pre:    0 <= i <= i+n <= a.length, x is an int,
//         a[i..i+n) is in ascending order. (In Dafny we would
//         write a[i..i+n] instead of a[i..i+n).)
// Eftir:  i <= k <= i+n.
//         a[i..k) <= x < a[k..i+n).
//         The array a is unchanged.
static int find( int[] a, int i, int n, int x )
{
  …
}
```

Svar/Answer:

Hluti II – Quicksort o.fl.
Part II – Quicksort etc.

Svarið að minnsta kosti einni spurningu í þessum hluta – Munið að svara a.m.k. 8 spurningum í heild
Answer at least one question in this part – Remember to answer at least 8 questions in total

4.

Gerið ráð fyrir að til sé Dafny fall með eftirfarandi lýsingu:
Assume a Dafny function with the following description:

```
method Partition( a: multiset<int> )
     returns( b: multiset<int>, c: seq<int>, d: multiset<int> )
   requires |a| > 0;
   ensures exists p | p in a ::
        (forall z | z in b :: z <= p) &&
        (forall z | z in c :: z == p) &&
        (forall z | z in d :: z >= p);
    ensures a == b+multiset(c)+d;
    ensures |c| > 0;
```

Skrifið Quicksort fall sem notar þetta fall sem hjálparfall til að varpa poka (multiset) yfir í raðaða runu sömu gilda. Ekki forrita Partition fallið hér.
Write a Quicksort function that uses this function as a helper function to transform a multiset into a sorted sequence of the same values. Do not program the Partition function here.

Svar/Answer:

5.

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

Svar/Answer:

6.

Forritið eftirfarandi Dafny fall – Program the following Dafny function:

```
method Quickselect( a: multiset<int>, k: int )
  returns( b: multiset<int>, c: seq<int>, d: multiset<int> )
    requires 0 <= k < |a|;
    ensures a == b+multiset(c)+d;
    ensures |b| <= k < |b|+|c|;
    ensures forall p,q | p in c && q in c :: p == q;
    ensures forall p,q | p in b && q in c :: p <= q;
    ensures forall p,q | p in d && q in c :: p >= q;
```

Þið megið nota Partition fallið sem hjálparfall.
You may use the Partition function as a helper function.

Svar/Answer:

Hluti III – Tvíleitartré
Part III – Binary Search Trees

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.
Answer at least two questions in this part – Remember to answer at least 8 questions in total
At the end of the exam there are descriptions of the most relevant functions in the file 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 aftasta hnút í milliröð sem inniheldur neikvæða tölu (tölu minni 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.

Assume the definition

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

as in our file BST.dfy. Also assume the functions IsTreePath, TreeSeq, PreSeq, MidSeq, PostSeq, PreSeqIncluding, etc.
Write a function that searches in a binary search tree and returns a reference to the rightmost node that contains a negative number (a number less than zero), or returns BSTEmpty is such a node does not exist in the tree.
Remember to write a full description with requires/ensures and to write an invariant for your loops if you use loops.

Svar/Answer:

8. **Leysið sama vandamál og að ofan, en núna í Java. Notið gamalkunna skilgreiningu á trjáhnútum, sem sjá má fyrir neðan. (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.)**

```
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; }
}
```

Solve the same problem as above, but this time in Java. Use our familiar definition of tree nodes, as seen above.
(I am skipping Usage/Pre/Post to save space because they are familiar and obvious. Note that this does not mean that students can skip writing familiar and obvious descriptions for their classes.)

Svar/Answer:

9. **Skrifið fall í Dafny sem tekur tvö viðföng, tvíleitartré t og heiltölu x og skilar true ef talan x er til í trénu en skilar false annars. Þið megið nota lykkju eða endurkvæmni, að því tilskildu að rökstuðningur sé réttur (þ.e. rétt requires, ensures og invariant).**
Write a function in Dafny that takes two arguments, a binary search tree t and an integer x and returns true if the number x exists in the tree and returns false otherwise. You may use a loop or recursion, given that the reasoning is correct (i.e. correct requires, ensures and invariant).

Svar/Answer:

10.

Leysið sama vandamál og að ofan, en í Java.

Solve the same problem as above, but in Java.

Svar/Answer:

Hluti IV – Ýmislegt
Part IV -- Miscellaneous

Svarið að minnsta kosti einni spurningu í þessum hluta – Munið að svara a.m.k. 8 spurningum í heild
Answer at least one question in this part – Remember to answer at least 8 questions in total

11.

Forritið stofninn á fallinu að neðan. Notið endurkvæmni en ekki lykkju.
Program the body of the function below. Use recursion and not a loop.

```
method Root( f: real->real, a: real, b: real, eps: real )
  returns( c: real, d: real )
    decreases ((b-a)/eps).Floor;
    requires a < b;
    requires eps > 0.0;
    requires f(a)*f(b) <= 0.0;
    ensures a <= c < d <= b;
    ensures d-c < eps;
    ensures f(c)*f(d) <= 0.0;
```

Vísbending: Eftirfarandi hjálparsetningu má sanna í Dafny. Þið þurfið ekki að kalla á hana.
Hint: the following lemma can be proven in Dafny. You do not need to call it.

```
lemma BisectionTermination( a: real, b: real, eps: real )
    requires b-a >= eps > 0.0;
    ensures ((b-a)/eps).Floor > ((b-a)/2.0/eps).Floor;
```

Svar/Answer:

12.

Forritið stofninn á fallinu að neðan. Notið lykkju en ekki endurkvæmni. Munið að setja decreases klausu í lykkjuna. Íhugið vísbendinguna í dæminu á undan.
Program the body of the function below. Use a loop and not recursion. Remember to put a decreases clause in the loop. Consider the hint in the previous question.

```
method Root( f: real->real, a: real, b: real, eps: real )
  returns( c: real, d: real )
    requires a < b;
    requires eps > 0.0;
    requires f(a)*f(b) <= 0.0;
    ensures a <= c < d <= b;
    ensures d-c < eps;
    ensures f(c)*f(d) <= 0.0;
```

Svar/Answer:

(auð blaðsíða/empty page)

(auð blaðsíða/empty page)

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
//         millirröð, 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 millirröð,
//         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)
```