

Lokapróf í Tölvunarfræði 1 (TÖL101G)

13. desember 2011 kl. 9:00

Nafn og kennitala

Dæmi	1	2	3	4	5	6	7	8	9	10	11 (auka)	Samtals
Stig												
af	15	13	7	10	5	5	5	10	15	15	10	100

Prófið er 15 tölusettar blaðsíður. Síðasta blaðsíðan er með lista yfir föll og klasa sem má nota í lausnum á forritum, ekki þarf að skila þeirri blaðsíðu með prófinu. **Engin hjálpargögn eru leyfileg.**

Athugið að öll föll verða að hafa lýsingu með „**Notkun:...**“, „**Fyrir:...**“ og „**Eftir:...**“. Allir klasar verða að hafa fastayrðingu gagna.

The exam is 15 numbered pages. The last page has a list of methods and classes you can use in your solutions, you do not need to hand in the last page with your exam. **No help materials are allowed.**

Note that all functions must have a description with „**Use:...**“, „**Pre:...**“ og „**Post:...**“. All classes must have a data invariant.

1. Fjölvalsspurningar (15 heildarstig). Dragið hring utan um rétt svar.

Multiple choice (15 points). Circle the correct answer.

(a) (2 stig) Hver af eftirfarandi `boolean` segðum skilar ekki alltaf `true` þegar strengirnir `s1` og `s2` innihalda sama gildi.

Which of the following `boolean` expressions does not always return `true` when the strings `s1` og `s2` contain the same value.

- i. `s1.equals(s2)`
- ii. `s1.length() == s2.length()`
- iii. `s1 == s2`
- iv. `s1.substring(0, s2.length()).equals(s2.substring(0, s1.length()))`
- v. `s1.length() + s2.length() >= 0`

(b) (2 stig) Hvaða lýsing á best við `s` að lokinni `keyrslu`.

Which of the following describes `s` best after running the code.

```
int s = 0;
for (int i = 1; i < a.length; i += 2) {
    if (a[i] > 0) {
        s += 1;
    }
}
```

- i. `s` er summa allra jákvæðra staka í `a`
`s` is the sum of all positive entries in `a`
- ii. `s` er summa allra jákvæðra staka í oddatölusætum í `a`
`s` is the sum of all positive entries in odd numbered indices of `a`
- iii. `s` er fjöldi jákvæðra staka í `a`
`s` is the number of positive entries in `a`
- iv. `s` er fjöldi jákvæðra staka í oddatölusætum í `a`
`s` is the number of positive entries in odd numbered indices of `a`

(c) (2 stig) Ef x er breytta af taginu `int` og y er af taginu `double`, hver af þessum gildisveitingum er ólögleg? – If x is a variable of type `int` and y is a variable of type `double`, which of these assignment statements is illegal?

- i. `y = x`
- ii. `y = (double) x`
- iii. `x = y`
- iv. `x = (int) y`
- v. `y = (int) x`

(d) (2 stig) Veljið þær staðhæfingar sem passa best inn í eyðurnar. “Samhliða forritun getur gefið en við verðum að passa okkur á .

Choose the statements that best fit in the blanks “Paralell programming can yield , but we must watch out for .

- i. hraðari þræði, öðrum forriturum
 faster threads, other programmers
- ii. hraðara forrit, sameiginlegum gögnum þráða
 a faster program, shared data between threads
- iii. betri kóða, samstillingu þráða
 better code, coordination of threads
- iv. styttri kóða, flóknari gagnavinnslu
 shorter code, increased complexity in data processing

(e) (4 stig) Hvaða lýsing á best við heiltöluna k að lokinni keyrslu, miðað við að N sé af taginu `int` og $N \geq 1$.

Which of the following describes the integer k best after running the code. Assume that N has type `int` and $N \geq 1$

```
int k = 0;
while (N > 1) {
    N = N/2;
    k++;
}
```

- i. k er $\log_2(N)$ – k is $\log_2(N)$
- ii. k er $\log_2(N)$ námundað að næstu heiltölu – k is $\log_2(N)$ rounded to the nearest integer
- iii. k er $\log_2(N)$ námundað niður á við – k is $\log_2(N)$ rounded down to the nearest integer
- iv. k er $\log_2(N)$ námundað upp á við – k is $\log_2(N)$ rounded up to the nearest integer

(f) (3 stig) Tölurnar $0, 1, 2, 3, 4, 5, 6$ eru settar á stafla í þessari röð með push aðgerðum í bland við pop aðgerðir. Alls eru framkvæmdar 7 push aðgerðir og 7 pop aðgerðir í einhverri röð og hver pop aðgerð prentar út gildið sem er skilað. Hvaða röð getur **ekki** prentast út?

The numbers $0, 1, 2, 3, 4, 5, 6$ are put on a stack in this order with push operations mixed with pop operations. A total of 7 push operations and 7 pop operations are performed in some order and each pop operation will print the value returned. Which of the following permutations **cannot** occur?

- i. 6 5 4 3 2 1 0
- ii. 0 1 4 3 2 5 6
- iii. 0 1 4 3 2 6 5
- iv. 3 2 4 6 0 1 5
- v. 1 2 3 0 4 6 5

2. (13 stig) 1 stig fyrir rétt svar, $-\frac{1}{2}$ fyrir rangt, 1 point for correct answer, $-\frac{1}{2}$ for incorrect.

Gerið ráð fyrir lykkju af eftirfarandi gerð, þar sem skilyrðið R hefur engar hliðarverkanir.

Assume the following loop structure where the condition R has no side effect.

```
// F
while (R)
{
    // I
    S
}
// E
```

Hverjar eftirfarandi fullyrðinga **verða að** vera sannar ef lykkjan er vel forrituð, og hverjar **geta verið** ósannar. Merkið „satt (S)“ við þær fullyrðingar, sem *verða að vera sannar*, og „ósatt ($Ó$)“ við þær fullyrðingar, sem *mega undir einhverjum kringumstæðum vera ósannar (ekki endilega kringumstæður sem koma upp í þessum forritskafli)*.

Which of the following assertions **must be** true if the loop is well programmed, and which ones **can be** false. Mark the ones that *must be true* with "true (T)", and mark the ones that *may be false under some circumstances (not necessarily circumstances that can happen in this particular code segment)* as "false (F)".

- (a) ☐ $F \Rightarrow I$. (Ef F er satt þá er I satt — If F is true then I is true).
- (b) ☐ $\{F\}S\{I\}$. (Ef F er satt og S er framkvæmt þá er I satt eftir það — If F is true and S is executed then I is true after that).
- (c) ☐ $\{I\}S\{I\}$. (Ef I er satt og S er framkvæmt þá er I satt eftir það — If I is true and S is executed then I is true after that).
- (d) ☐ $\{I \wedge R\}S\{I\}$. (Ef I og R er sannar og S er framkvæmt þá er I satt eftir það — If I and R are true and S is executed then I is true after that).
- (e) ☐ $\{F \wedge R\}S\{I\}$. (Ef F og R eru sannar og S er framkvæmt þá er I satt eftir það — If F and R are true and S is executed then I is true after that).
- (f) ☐ $\neg I \Rightarrow E$. (Ef I er ósatt þá er E satt — If I is false then E is true).
- (g) ☐ $I \Rightarrow E$. (Ef I er satt þá er E satt — If I is true then E is true).
- (h) ☐ $I \wedge \neg R \Rightarrow E$. (Ef I er satt og R er ósatt þá er E satt — If I is true and R is false then E is true).
- (i) ☐ $I \wedge R \Rightarrow E$. (Ef I er satt og R er satt þá er E satt — If I is true and R is true then E is true).
- (j) ☐ I er satt rétt fyrir sérhverja umferð lykkjunnar. I is true just before each pass through the loop.
- (k) ☐ I er satt rétt eftir sérhverja umferð lykkjunnar. I is true just after each pass through the loop.
- (l) ☐ I er satt á sérhverjum tímapunkti meðan á umferð lykkju stendur. I is true at each instant while the loop is running.
- (m) ☐ Rétt fastayrðing lykkju tryggir ávallt að lykkjan taki enda. A correct loop invariant always ensures that the loop terminates.

3. (7 stig) Í klösunum A og B er aðferðin f yfirskrifuð og x, y eru jákvæðar heiltölur. Hvaða skilyrði verða að gilda um x og y til þess að rökræðilegt samband klasanna A og B sé rétt?
The method f is overwritten in classes A and B , and x, y are positive integers. What relationship must hold between x and y in order for the logical relationship between A and B is correct?

```
public class A {
    ...
    // Notkun: t = a.f(n)
    // Fyrir: n er margfeldi af 24
    // Eftir: t er margfeldi af 120
    // Use: t = a.f(n)
    // Pre: n is a multiple of 24
    // Post: t is a multiple of 120
    public int f(int n) { ... }
}

public class B extends A {
    ...
    // Notkun: t = b.f(n)
    // Fyrir: n er margfeldi af x
    // Eftir: t er margfeldi af y
    // Use: t = a.f(n)
    // Pre: n is a multiple of x
    // Post: t is a multiple of y
    public int f(int n) { ... }
}
```

4. (10 stig) Eftirfarandi útfærsla á quicksort raðar fylki af heiltölum í vaxandi röð og notar hjálparfallið `skipta`.

The following implementation of quicksort sorts an array of integers in increasing order and uses the function `skipta`.

```
// Notkun: quicksort(a,i,j)
// Fyrir: a[i],...,a[j] er löglegt svæði í a
// Eftir: svæðið a[i],...,a[j] er raðað í vaxandi röð
public static void quicksort(int[] a, int i, int j) {
    if (i >= j) return;
    int k = skipta(a,i,j);
    quicksort(a,i,k-1);
    quicksort(a,k+1,j);
}
```

Fyllið inn í fyrir og eftirskilyrðin fyrir `skipta` til að hægt sé að sanna quicksort fallið.

Fill in the blanks in the Pre and Post conditions for the function `skipta` so that the quicksort function can be proven correct.

```
// Notkun/Use: k = skipta(a,i,j)
// Fyrir/Pre:
//
//
// Eftir/Post:
//
//
//
public static int skipta(int[] a, int i, int j) { ... }
```

5. (5 stig) Heiltölurnar 1-10 eru settar inn í tvíleitartré sem lyklar. Þegar við leitum að 8 eru lyklarnir 5,7,9,8 skoðaðir í þessari röð. Þegar við leitum að 3 eru lyklarnir 5,2,4,3 skoðaðir í þessari röð. Teiknið tréð. (Vísbending: allir lyklarnir 1-10 eru í trénu og það er aðeins ein möguleg lausn).

The integers 1-10 are inserted in a binary search tree as keys. When we search for 8 we examine the keys 5,7,9,8 in this order. When we search for 3 we examine 5,2,4,3 in this order. Draw the tree. (Hint: all of the keys 1-10 are in the tree and there is only one solution).

6. (5 stig) `BuggySum(0, a.length-1)` ætti að reikna summuna af tölunum í fylkinu `a`, en gerir það ekki. Finnið villuna, útskýrið af hverju hún kemur fyrir og sýnið hvernig má laga forritið til að það virki rétt.

`BuggySum(0, a.length-1)` should compute the sum of numbers in the array `a`, but it doesn't. Find the bug, explain why it happens and show how to fix the program so it works correctly.

```
public static double BuggySum(double[] a, int k) {  
    if (k == 0) return 0.0;  
    else return a[k] + BuggySum(a, k-1);  
}
```

7. (5 stig) Hve langan tíma tekur að keyra kóðann í línunum 1-5 að neðan. Gefið svarið með $O()$ rithætti og rökstyðjið svarið. – What is the running time of the code in lines 1-5 below. Give the answer using $O()$ notation and justify your answer.

```
public static double array_max(int[] a, int i) {  
    double max = 0.0;  
    for (int k = 0; k < i; k++) {  
        if (max < a[k]) {  
            max = a[k];  
        }  
    }  
    return max;  
}
```

```
1 int N = a.length;  
2 int[] c = new int[N+1];  
3 for (int i = 0; i <= N; i++) {  
4     c[i] = array_max(a, i);  
5 }
```

8. (10 stig) Skrifðu fall sem tekur inn raðað fylki af heiltölum og telur hve margar ólíkar tölur koma fyrir oftast en einu sinni. Dæmi: Fyrir inntakið $\{0, 1, 1, 1, 2, 4, 4, 5, 6, 6\}$ er svarið 3, því þrjár tölur (1, 4 og 6) koma fyrir tvisvar eða oftast. Skrifðu **fastayrðingu** **lykkju** í forritinu ykkar.

Write a function that takes as input a sorted array of integers and counts the number of distinct integers that appear at least once. Example: For the input $\{0, 1, 1, 1, 2, 4, 4, 5, 6, 6\}$ the answer is 3, since 3 numbers (1, 4, and 6) appear more than once. Write a **loop invariant** in your program.


```
}
```

```
// Notkun: m = s.mean()  
// Fyrir: s er ekki tómur  
// Eftir m er meðaltalið af gögnunum í s  
// Use: m = s.mean()  
// Pre: s is not empty  
// Post: m is the mean of the data in s  
public double mean() {
```

```
}
```

```
// Notkun: var = s.variance()  
// Fyrir: s er ekki tómur  
// Eftir var er kvaðratfrávikðið af gögnunum í s  
// Use: var = s.variance()  
// Pre: s er ekki tómur  
// Post: var is the variance of the data in s  
public double variance() {
```

```
}
```

```
}
```

10. (15 stig) Í þessu dæmi eru tvær aðferðir við að leita að hlutstrengjum í lengri streng. Strengurinn s er af lengd N og a er fylki af M strengjum, þar sem allir strengir í a eru af lengd k . Við gerum ráð fyrir því að k sé lítil tala (í mesta lagi 50) og því taka `equals` og `compareTo` allar $O(1)$ tíma fyrir hvern af strengjunum í a . Athugið að keyrslutíminn í liðum a. og b. er því fall af N og M .

This question has two methods for searching for substrings in a larger string. The string s has length N and a is an array of M strings, where each string in a has length k . We will assume that k is a small number (at most 50) so that the `equals` and `compareTo` methods take $O(1)$ time for each of the strings in a . Note that the running time in parts a. and b. is a function of both N and M .

- (a) Hver er keyrslutíminn fyrir fallið `simplesearch`?

What is the running time for the method `simplesearch`?

```
1  public static int simplesearch(String s, String[] a, int k) {
2      int count = 0;
3      for (int i = 0; i < s.length()-k; i++) {
4          String sub = s.substring(i,i+k);
5          for (int j = 0; j < a.length; j++) {
6              if (a[j].equals(sub)) {
7                  count++; break;
8              }
9          }
10     }
11     return count;
12 }
```

- (b) Hver er keyrslutíminn fyrir fallið `bettersearch`?

What is the runningtime for the method `bettersearch`?

```
1  public static int bettersearch(String s, String[] a, int k) {
2      String[] b = new String[s.length()-k+1];
3      for (int i = 0; i < s.length()-k+1; i++) {
4          b[i] = s.substring(i,i+k);
5      }
6      Arrays.sort(b);
7
8      int count = 0;
9      for (int j = 0; j < a.length; j++){
10         if (binarySearch(b, 0, b.length,a[j]-1) >= 0) {
11             count++;
12         }
13     }
14     return count;
15 }
```

(c) Forritið `binarySearch` fallið skv. eftirfarandi lýsingu

Program the `binarySearch` method using the following description

```
// Notkun: k = binarySearch(a,i,j,x)
// Fyrir: a er raðað í vaxandi röð
// Eftir: k er -1 ef x er ekki í a
//        annars er k þ.a. a[k] == x
// Use: k = binarySearch(a,x,i,j)
// Pre: a is in increasing order
// Post: k is -1 if x is not in a
//        otherwise k is s.t. a[k] == x
public static int binarySearch(String[] a, int i, int j, String x)
```

11. (10 stig) Aukaspurning. Ekki reyna við þessa spurningu fyrr en þið hafið svarað öllum öðrum spurningum á prófinu.

Extra question. Do not attempt to solve this question until you have answered all other questions on the exam.

Jólasveinninn geymir allar jólagjafir til barna í birgðastöð á norðurlandinu. Hver gjöf er merkt með nafni þess barns sem á að fá hana. Undanfarin ár hefur sveinki notast við einfalt kerfi til að hafa uppi á gjöfum. Til að ná í gjöf fyrir Sigga er reiknað út hakkafallið $h(\text{"Siggi"})$ sem vísar á hólfið í birgðastöðinni. Þá er kíkt á það hólfi hvort og merkimiðarnir bornir saman, ef gjöfin hans Sigga er ekki í því hólfi er farið í næsta hólfi til hægri og kíkt á þann miða og koll af kolli þar til annað hvort gjöfin finnst eða við komum að tómu hólfi (og þá fær Siggi enga gjöf).

En fyrir þessi jól tók nýr álfur við stjórninni í birgðastöðinni og kom með nýtt kerfi til að setja gjafir í hólfi. Birgðastöðinni er skipt í tvo jafnstóra helminga og í stað eins hakkafalls er nú notast við tvö hakkaföll. Ef við köllum helmingana $T1$ og $T2$ og hakkaföllin $h1$ og $h2$ þá er notast við eftirfarandi lýsingu til að setja gjafir á sinn stað. T.d. ef setja á inn gjöfina fyrir Gunnu er reiknað út $h1(\text{"Gunna"})$ og farið í það hólfi í $T1$, ef það er tomt er gjöfin sett þar inn. Ef hólfið er ekki tomt er sú gjöf sem er fyrir, segjum gjöfin hans Sigga, tekin úr hólfinu í $T1$, gjöfin hennar Gunnu sett inn og gjöfin hans Sigga færð í hólfið $h2(\text{"Siggi"})$ í $T2$. Ef það hólfi er ekki tomt er gjöfin hans Sigga sett í hólfið og gjöfin sem þar er fyrir tekin út og færð í sitt hólfi í $T1$ o.s.frv þangað til að einhver gjöf endar í tómu hólfi (þessi aðferð gæti lent í óendanlegri lykkju, en við ætlum ekki að hafa áhyggjur af því).

Santa Claus stores all Christmas presents for children in a depot on the North Pole. Each gift has a label with the name of the child which will receive the gift. In recent years Santa has used a simple system of locating gifts. To find the gift for Siggi we compute the hash function $h(\text{"Siggi"})$ which will point to a slot in the depot. We then look into that slot and compare the labels, if Siggi's gift is not in that slot we look at the next slot to the right and compare labels and so on, until either we find Siggi's gift or we find an empty slot (in which case Siggi doesn't get a gift this Christmas).

But for this christmas a new elf took charge of the depot and changed to new system of inserting and locating gifts. The depot is now split into two equally sized parts and instead of just one hash function we have two. Let $T1$ and $T2$ be the two parts of the depot and $h1$ and $h2$ be the two hash functions. We use the following method to insert presents into slots. If we insert a gift for Gunna we compute $h1(\text{"Gunna"})$ and locate that slot in $T1$, if it is empty we insert Gunna's present there. If the slot is not empty we take the gift in that slot, say Siggi's gift, and insert Gunna's gift into the slot. Siggi's gift is then inserted into $h2(\text{"Siggi"})$ in $T2$. If that slot is not empty we replace it with Siggi's gift and insert that gift into the correct slot in $T1$ and so on and so forth until a gift winds up in an empty slot (this could get stuck in an infinite recursion, but we won't worry about that).

- (a) Hver er besti og versti keyrslutími fyrir leit (get) og innsetningu (put) fyrir þessa aðferð, rökstyðjið og skilið svari með $O()$ rithætti.

What is the best and worst case running time for search (get) and insertion (put) for this system. Justify your answer and write using $O()$ notation.

- (b) Forritið `put(key,value)` aðferðina miðað við að hakkaföllin $h1, h2$ séu skilgreind og $T1, T2$ séu jafnstór fylki af réttu tagi.

Program the `put(key,value)` method assuming the hashfunctions $h1, h2$ are defined and $T1, T2$ are equally sized arrays of the correct type.

Þessu blaði þarf ekki að skila með próflausninni. – You do not need to turn in this sheet with your solution.

- Math

- `double abs(double a)` absolute value of a
- `double max(double a, double b)` maximum of a and b
- `double min(double a, double b)` minimum of a and b
- `double pow(double a, double b)` raise a to the b-th power (a^b)
- `double log(double a)` natural logarithm of a
- `double exp(double a)` exponential (e^a)
- `double sqrt(double a)` square root of a

- String

- `int length()` string length
- `char charAt(int i)` i-th character
- `String substring(int i, int j)` i-th through (j-1)-st characters
- `int compareTo(String t)` comparison, returns a value < 0 if this string is before t in alphabetical order, returns > 0 if t is before this and returns 0 if both strings are identical
- `boolean equals(String t)` returns true if this has the same value as t

- Vector<E>

- `boolean add(E o)`: appends the specified element to the end of this list.
- `E get(int index)`: returns the element at the specified position.
- `int indexOf(Object o)`: returns index of the first occurrence of element, or -1 if it's not there.
- `boolean isEmpty()`: returns true if this list contains no elements.
- `E set(int index, E element)`: replaces element at the specified position (returns old element).
- `int size()`: returns the number of elements in this list.