http://kodu.ut.ee/~kiho/ads/Praktikum/

2. KMB Kombinatoorika. Rekursioon

Mõisteid

Rekursiivne meetod

Permutatsioonid

Kombinatsioonid

Alamhulgad

Java generaator-klass

Variantide läbivaatamine

Harjutusülesanded

KMB_1. Koostada rekursiivne meetod antud järjendis esinevate negatiivsete elementide loendamiseks.

KMB_2-0. Koostada generaatorklass massiivina antud täisarvujärjendist *k*-kaupa kombinatsioonide genereerimiseks, kui abigeneraatorina kasutatakse klassi *Gen_IndKombin*.

KMB_2-1. Ülesanne 1.1* (Kõik suurima elementide arvuga nullsummalised osajärjendid.)

KMB_3. Ülesanne 1.4* (Kes liftiga, kes trepist üles.)

KMB_4. Ülesannete kogust ülesanne nr 4.29a. Kast kirjeldada klassi *Kast.java* isendina (vt *KastMall.java*).

KMB_5. Koostada rekursiivne meetod leidmaks kõik *n* lipu „rahumeelsed“ paigutused *n*x*n* malelaual, st sellised paigutused, kus ükski lipp ei ole teise lipu tules

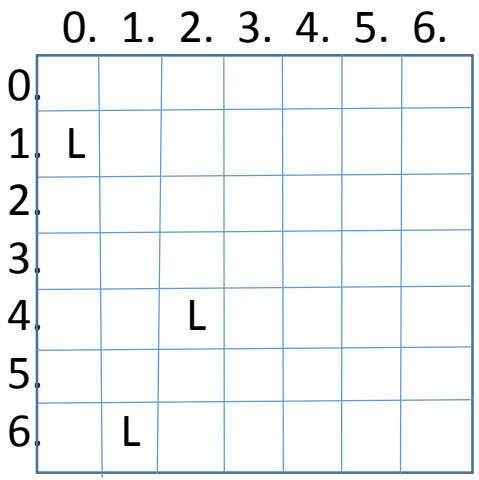
(a) seisude generaatorit kasutades;

(b) generaatorit kasutamata.

(Suuniseid järgmistel slaididel.)

* http://kodu.ut.ee/~kiho/ads/Praktikum/Arvutipraktikum/2-KMB/%C3%9Clesanded/Peafail.pdf

Lippude rahumeelse paigutuse korral saab ühes veerus (ühel liinil) olla vaid üks lipp. Vajaliku seisu otsimisel asetame lippe liinidele järjekorras vasakult paremale (veeruindeksi kasvamise järjekorras) ruutudele, mis ei ole ühegi eelnevalt asetatud lipu tules.
Jooksvat seisu, kui on paigutatud *k* lippu, iseloomustab nende lippude reaindeksite list
js = [*r*0, *r*1, …, r(*k*-1)].
Sellest saame järgmise seisu, lisades järgmise lipu veergu indeksiga *k* -- ruudule, mis pole ühegi eelmise lipu tules.
Näiteks *n*=7 korral iseloomustab kolme lipu (*k* = 3) ühte võimalikku seisu

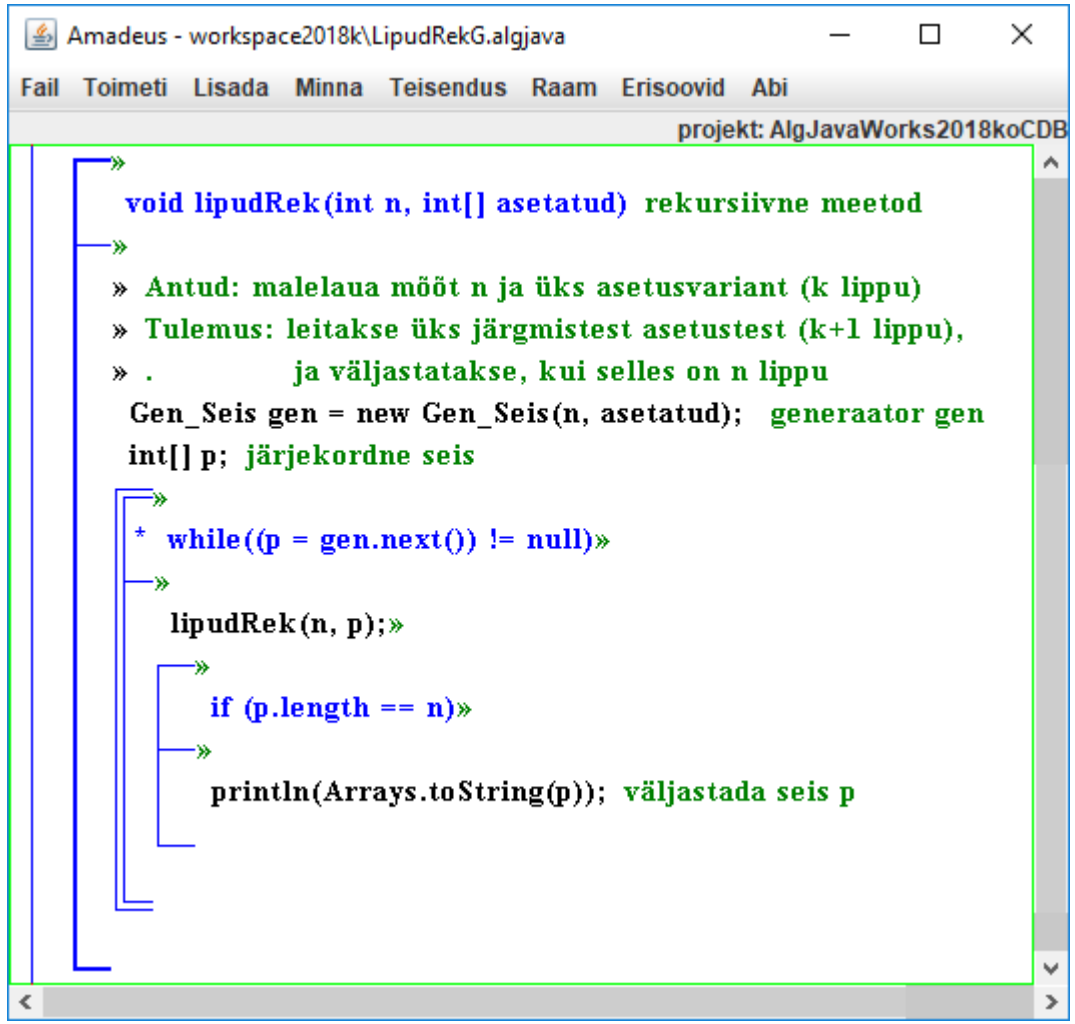


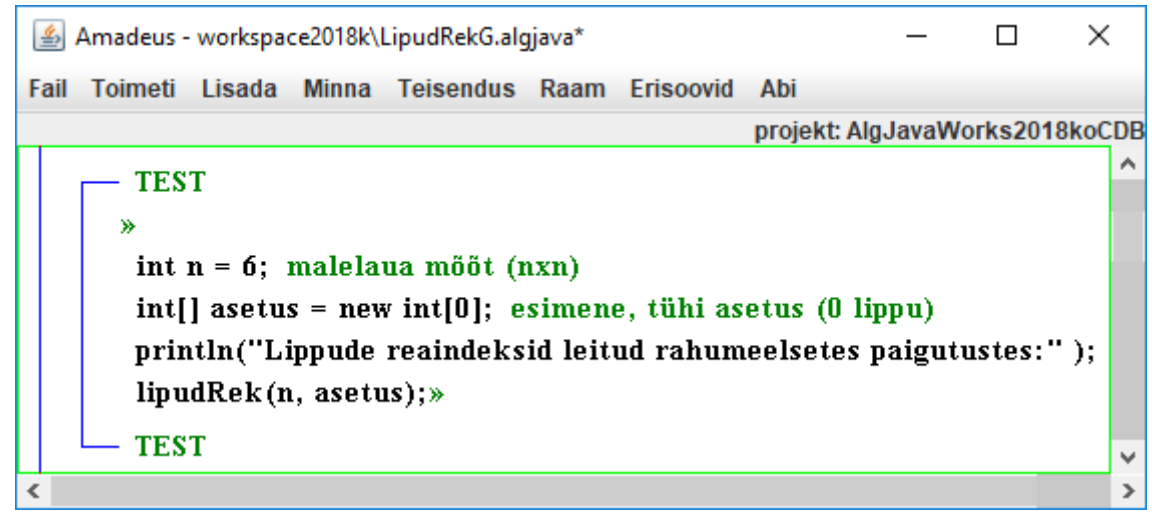
list [1, 6, 4].
Ühte järgmistest seisudest (neli lippu) kirjeldab list [1, 6, 4, 0]. Ja veel ühte -- list [1, 6, 4, 2].

Ülesande **KMB_5a** lahendamine.

Koostame generaator-klassi (vt *Gen_SeisMall.java*), milles
konstruktorile antakse ette malelaua mõõt *n* ja
jooksev seis *js* – mõnede esimeste lippude paigutus (nende reanumbrite listina);
meetodi *next*() tagastustüübiks on *int*[],
järjekordsel rakendamisel antakse massiivina välja
järjekordne uus seis, mis on saadud antud jooksvale seisule *js*
ühe sobiva reanumbri (järgmise lipu asukoha) lisamise teel,
või *null*, kui järgmise lipu asukohta enam leita.

Seejärel saame koostada nõutud rekursiivse meetodi, vt skeem järgmisel slaidil.


Amadeus - workspace2018k\LipudRekG.algjava
Fail Toimeti Lisada Minna Teisendus Raam Erisoovid Abi
projekt: AlgJavaWorks2018koCDB
»
void lipudRek(int n, int[] asetatud) rekursiivne meetod
»
» Antud: malelaua mõõt n ja üks asetusvariant (k lippu)
» Tulemus: leitakse üks järgmistest asetustest (k+1 lippu),
» . ja väljastatakse, kui selles on n lippu
Gen_Seis gen = new Gen_Seis(n, asetatud); generaator gen
int[] p; järjekordne seis
»
* while((p = gen.next()) != null)»
»
lipudRek(n, p);»
»
if (p.length == n)»
»
println(Arrays.toString(p)); väljastada seis p


Amadeus - workspace2018k\LipudRekG.algjava*
Fail Toimeti Lisada Minna Teisendus Raam Erisoovid Abi
projekt: AlgJavaWorks2018koCDB
TEST
»
int n = 6; malelaua mõõt (nxn)
int[] asetus = new int[0]; esimene, tühi asetus (0 lippu)
println("Lippude reaindeksid leitud rahumeelsetes paigutustes:");
lipudRek(n, asetus);»
TEST

KMB_6. Koostada meetod antud järjendist (ühe) pikima monotoonse osajärjendi leidmiseks.
Järjend on monotoonne, kui see on kas mittekahanev või mittekasvav.

Näiteid:

```
Järjend a:     [52, 38, 46, 14, 21, 55, 18, 68, 21, 15, 51, 82, 77, 91, 25]
Järjendi a pikim monotoonne osajärjend:        [38, 46, 55, 68, 82, 91]

Järjend a:     [44, 16, 99, 45, 81, 90, 73, 70, 55, 67, 33, 57, 41, 90, 86]
Järjendi a pikim monotoonne osajärjend:        [99, 81, 73, 70, 67, 57, 41]
```

```
»
 ArrayList<Integer> pikimMonotOsaj(ArrayList<Integer> a)»
»
» Antud: täisarvude järjend a listina
» Tulemus: tagastatakse a pikim monotoonne osajärjend
 int n = a.size();»
 ArrayList<Integer> c = null;  // tulemuse koht
» järjendi a osajärjendite (ehk alamhulkade) läbivaatamine alates pikimast,
» st pikkuse mittekahanemise järjekorras, kuni leitakse monotoonne osajärjend
 kasutatakse kombinatsioonide generaatorit (kas Gen_Kombin või Gen_IndKombin)
return c;»
```

```
»
 boolean onMonot(ArrayList<Integer> a)  järjendi monotoonsuse kontroll
 Allikas: vt https://stackoverflow.com/questions/8778775/check-if-arraylist-is-sorted?lq=1
» Antud: täisarvude järjend a listina
» Tulemus: tagastatakse
» .              true, kui a on kas mittekahanev või mittekasvav;
» .              false vastasel korral (kui a ei ole mittekahanev ega mittekasvav)
 boolean mittekahanev = true, mittekasvav = true;»
»
* for (int i = 1; i < a.size() && ( mittekahanev || mittekasvav ); i++) »
»
  mittekahanev = mittekahanev && a.get(i) >= a.get(i-1);»
  mittekasvav = mittekasvav && a.get(i) <= a.get(i-1);»

return mittekahanev || mittekasvav;»
```