LUNDS TEKNISKA HÖGSKOLA Institutionen för datavetenskap

Kontrollskrivning 3, Programmeringsteknik för D/C

2015–01–12, 8.00–13.00

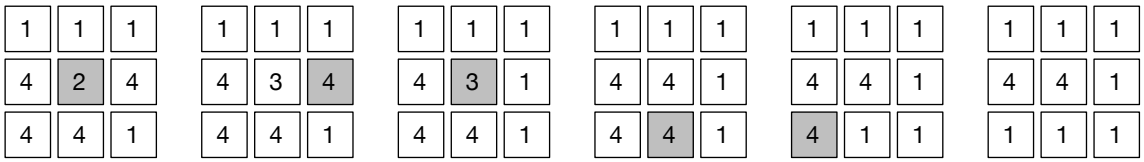
Anvisningar: Preliminärt ger uppgifterna $12 + 6 + 22 = 40$ poäng. Tillåtet hjälpmedel: Java-snabbreferens.

Omkontrollskrivning på hela kursen äger rum fredag 8 maj 8–13 i MA10. Det behövs inte någon föranmälan till omkontrollskrivningarna. Det är tillåtet att försöka höja sitt betyg (”plussa”) vid en omkontrollskrivning.

1. I spelet FourWay är spelplanen ett kvadratiskt rutnät med $n \times n$ rutor (n är alltid udda).[1] I varje ruta finns ett tal mellan 1 och 4. En omgång i spelet går till på följande sätt:

   1. Låt mittrutan vara aktuell ruta.
   2. Upprepa tills aktuell ruta är utanför rutnätet:
      - Om aktuell ruta innehåller talet 1 ändra 1-an till en 2-a och gå åt norr
      - Om aktuell ruta innehåller talet 2 ändra 2-an till en 3-a och gå åt öster
      - Om aktuell ruta innehåller talet 3 ändra 3-an till en 4-a och gå åt söder
      - Om aktuell ruta innehåller talet 4 ändra 4-an till en 1-a och gå åt väster

   Exempel på en spelomgång när $n = 3$ (den aktuella rutan är markerad med grå färg):

   Spelet beskrivs i följande klass (klassen har fler metoder, men dem ska du inte implementera):

   FourWay

   ```
   /** Skapar ett spel med n*n rutor med 1-or i alla rutor */
   FourWay(int n);

   /** Genomför en omgång av spelet */
   void oneRound();

   /** Returnerar true om alla rutor innehåller 1-or, annars false */
   boolean allOnes();
   ```

   Implementera klassen. Spelplanen bör naturligtvis vara en heltalsmatris.

2. Ordbehandlingsprogram som kan avstava ord använder dels algoritmer som beskriver reglerna för avstavning, dels listor med ord där avstavningarna inte följer reglerna. Den enklaste regeln för avstavning av svenska ord är:

   > Bindestreck får placeras mellan bokstäver på så sätt att efter varje bindestreck följer en konsonant och en vokal.

   Skriv en metod som implementerar avstavning enligt denna regel. Givet ett ord (en sträng innehållande bokstäverna a–ö) ska metoden returnera en vektor med de positioner i ordet efter vilka ordet kan avstavas. Vektorn ska ha lika många element som antalet avstavningspositioner. Du får använda en färdigskriven metod med anropet `isVowel(ch)` som ger `true` om tecknet `ch` är en vokal, `false` annars. Metoden ska ha följande rubrik:

   ```
   public static int[] hyphenate(String word);
   ```

   Exempel:

   algoritmer $\rightarrow \{1, 3, 6\}$ (dvs al-go-rit-mer)
   ordbehandlingsprogram $\rightarrow \{2, 4, 8, 14, 17\}$ (ord-be-hand-lingsp-rog-ram)
   katt $\rightarrow \{\}$ (katt)

[1] Här beskrivs bara en del av spelet; i verkligheten är det mera komplicerat.

3. ”Bulgarisk patiens” är ett läggspel med enkla regler:[2]

   - Tag N kort, fördela dem slumpmässigt på ett antal högar (1..N stycken, antalet väljs slumpmässigt). Det ska vara minst ett kort i varje hög. Vi förutsätter i fortsättningen att N är ett triangeltal, det vill säga att $N = 1 + 2 + \cdots + k$ för något k.
   - Upprepa tills antalet kort i högarna är $1, 2, \ldots, k$ i någon ordning:
     - Tag ett kort från varje hög, lägg dem i en ny hög.

   I nedanstående exempel visas först antalet kort i högarna när 10 kort fördelats på 6 högar. Sedan visas högarna efter varje drag och till sist antalet drag.

```
2 1 2 1 2 2
1 1 1 1 6
5 5
4 4 2
3 3 1 3
2 2 2 4
1 1 1 3 4
2 3 5
1 2 4 3
8 moves.
```

   Utskriften har genererats av följande program (med `n` = 10):

```
public class SolitaireTest {
    public static void main(String[] args) {
        System.out.print("Number of cards: ");
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        scan.close();
        BulgarianSolitaire bs = new BulgarianSolitaire(n);
        bs.print();
        int moves = 0;
        while (!bs.atGoal()) {
            bs.move();
            bs.print();
            moves++;
        }
        System.out.println(moves + " moves.");
    }
}
```

   Implementera klassen `BulgarianSolitaire`. Anvisningar:

   - Skriv en dokumentationskommentar (`/** ... */`) som talar om vad varje metod i klassen gör.
   - Använd den färdigskrivna klassen `Pile` för att representera en hög med kort. Klassen har följande specifikation:

```
Pile(int size);  // skapar en hög med size kort
int getSize();   // returnerar antalet kort i högen
void putCard();  // lägger ett kort i högen
void takeCard(); // tar ett kort från högen
```

   - Samla `Pile`-objekten i en `ArrayList`. Det får inte finnas några tomma högar i listan.

[2] `http://en.wikipedia.org/wiki/Bulgarian_solitaire`