Vejledende løsninger

Opgave 1

Spørgsmål 1.1

```
a = b - a;
b = b - a;
a = b + a;
```

Opgaven har flere løsninger. En anden løsning er:

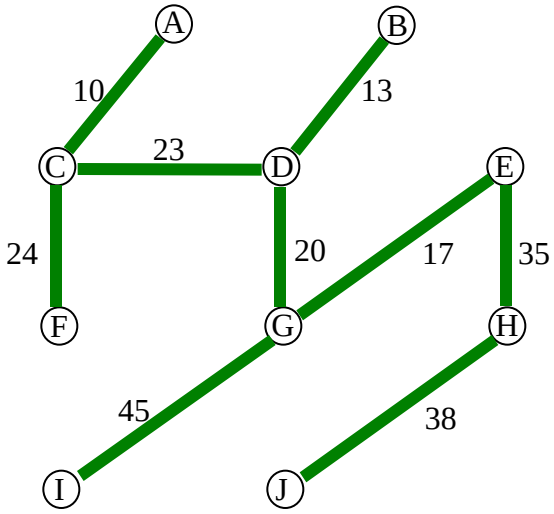
```
a = b + a;
b = a - b;
a = a - b;
```

Tilføjelse. Med følgende løsning undgås aritmetisk overløb:

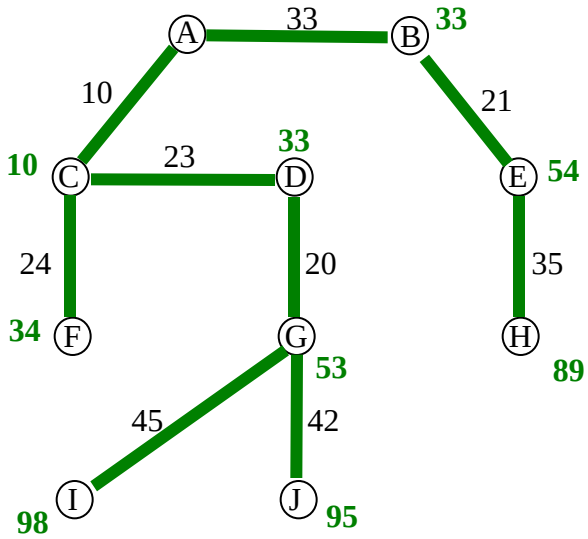
```
if ((a >= 0) == (b >= 0) { // a og b har samme fortegn
    a = b - a;
    b = b - a;
    a = b + a;
}
else { //a og b har forskelligt fortegn
    a = b + a;
    b = a - b;
    a = a - b;
}
```

Opgave 2

Spørgsmål 2.1



Spørgsmål 2.2



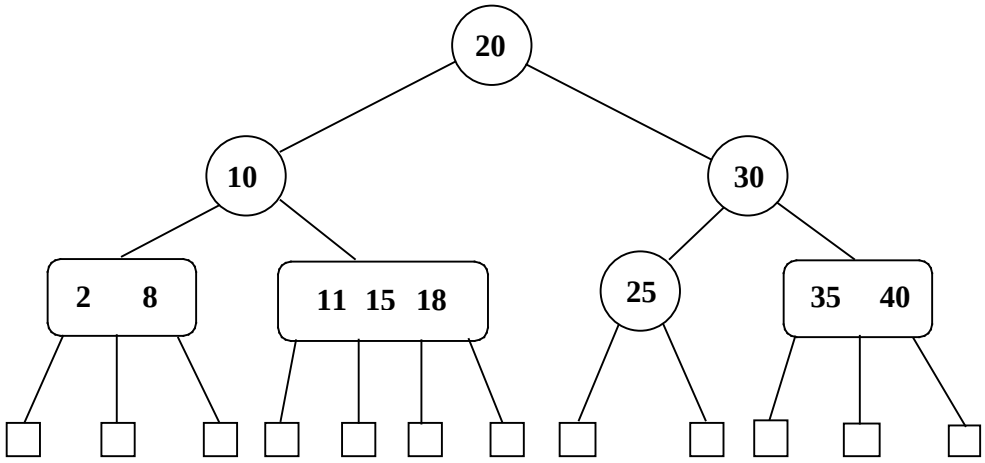
Opgave 3

Spørgsmål 3.1

(1) Træet er et binært søgetræ, idet
   a) enhver intern knude har to hægter (til sine to undertræer)
   b) for enhver intern knude gælder, at alle nøgler i det venstre undertræ er mindre end knudens nøgle, mens alle nøgler i det højre undertræ er større end (eller lig med) knudens nøgle.

(2) Træet opfylder rød-sort-betingelserne, idet
   a) ingen vej fra roden til en ekstern knude indeholder to røde kanter i træk
   b) enhver vej fra roden til en ekstern knude indeholder det samme antal sorte kanter, i dette tilfælde 3.

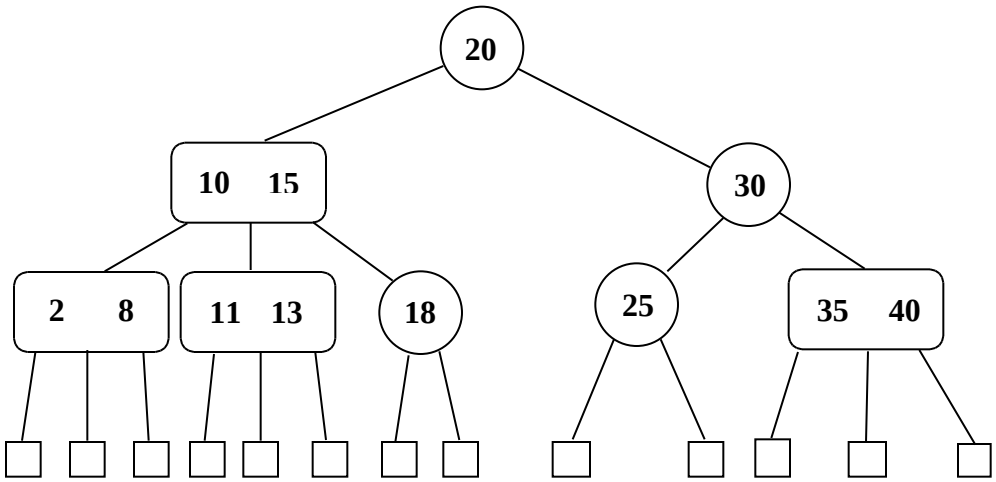
Spørgsmål 3.2



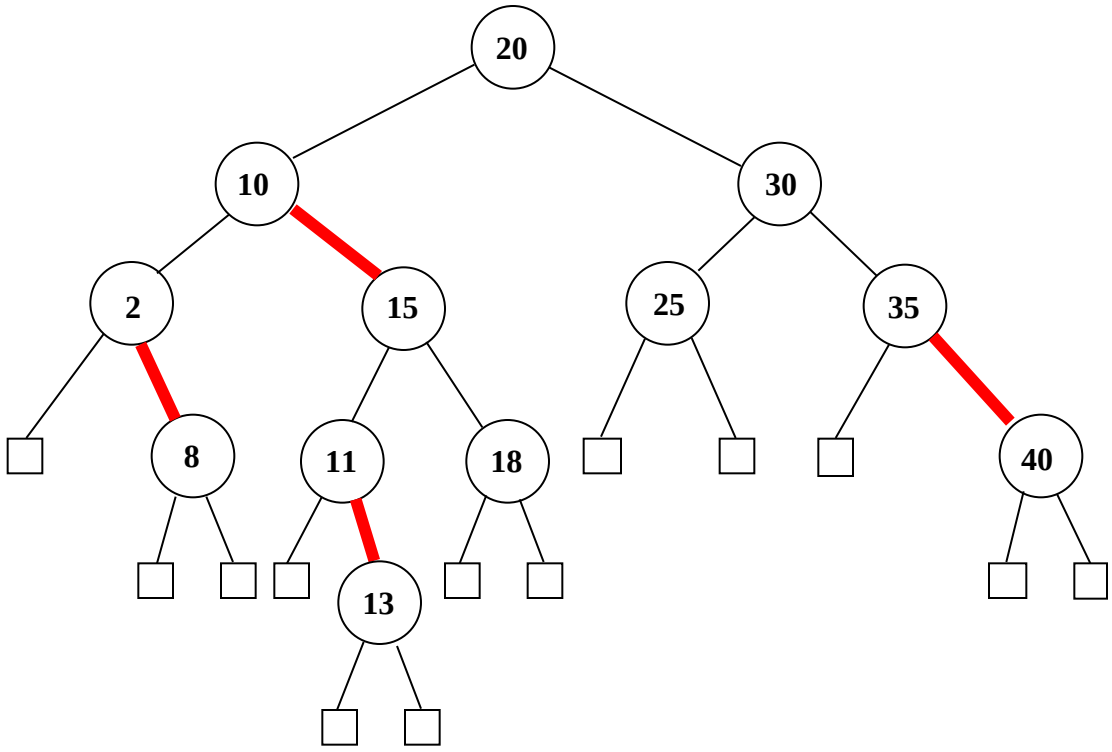
Spørgsmål 3.3

Opgaven løses lettest ved at indsætte nøglen i 2-3-4-træet fra spørgsmål 4.3, for derefter at konvertere det resulterende 2-3-4-træ til et rød-sort-træ.

(1) Nøglen 13 indsættes i 2-3-4-træet:



(2) Dette træ konverteres til et rød-sort-træ:



Opgave 4

Spørgsmål 4.1

```
/* A: */
           Bolt b = (Bolt) e;
           return size < b.size ? -1 : (size > b.size ? 1 : 0);
```

Spørgsmål 4.2

```
/* B: */
           b++;

/* C: */
           n++;

/* D: */
           n = i;
           while (nut[n].compareTo(bolt[b]) != 0)
               n++;

/* E: */
           b = i;
           while (bolt[b].compareTo(nut[n]) != 0)
               b++;
```

Spørgsmål 4.3

```
/* F: */
           while (e[i].compareTo(pivot) < 0)
               i++;
           while (e[j].compareTo(pivot) > 0)
               j--;
```

Spørgsmål 4.4

```
/* G: */
           partition(bolt, n, left, right);
           i = partition(nut, b, left, right);
           quicksort_nuts_and_bolts(left, i-1);
           quicksort_nuts_and_bolts(i+1, right);
```

Opgave 5

Spørgsmål 5.1

```
  private int karakter;
  Karakter(int k){karakter=k;}
```

Spørgsmål 5.2

```
  boolean lovlig(){
    return karakter==0||karakter==3||karakter==13||
      (karakter>=5&&karakter<=11);
  }
```

Spørgsmål 5.3

```
  public String toString(){ return
(karakter<5?"0"+karakter:""+karakter);}
```

Spørgsmål 5.4

```
  Karakter(String s){
    if(s.equals("00")||s.equals("0"))karakter=0;else
    if(s.equals("03")||s.equals("3"))karakter=3;else
    if(s.equals("5"))karakter=5;else
    if(s.equals("6"))karakter=6;else
    if(s.equals("7"))karakter=7;else
    if(s.equals("8"))karakter=8;else
    if(s.equals("9"))karakter=9;else
    if(s.equals("10"))karakter=10;else
    if(s.equals("11"))karakter=11;else
    if(s.equals("13"))karakter=13;
   }
```

Spørgsmål 5.5

```
   Karakter(Karakter k){this(k.toString());}
```

Spørgsmål 5.6

```
   void haev(){
     if(karakter==0)karakter=3;else
     if(karakter==3)karakter=5;else
     if(karakter==11)karakter=13;else
     if(karakter!=13)karakter++;
   }
```

Spørgsmål 5.7

```
   void haev(int i){while(i>0){i--;haev();}}
```

Opgave 6

Spørgsmål 6.1

```
  Bestaaet(boolean ok){this.ok=ok;}
```

Spørgsmål 6.2

```
  public String toString(){ return (ok?"Bestået":"Ikke
bestået");}
```

Spørgsmål 6.3

```
interface Resultat{
   public boolean erBestaaet();
}
I Bestaaet:
   public boolean erBestaaet(){return ok;};
I Karakter:
   public boolean erBestaaet(){return karakter>=6;};
}
```

Spørgsmål 6.4

```
        int bestaa=0;
        for(int i=0;i<tabel.length;i++)
            if(tabel[i].erBestaaet())bestaa++;
        System.out.println("bestået "+bestaa);
```

Spørgsmål 6.5

```
        int antal=0;
        for(int i=0;i<tabel.length;i++)
            if(tabel[i] instanceof Karakter)antal++;
        System.out.println("karakter "+antal);
```

Spørgsmål 6.6

```
        int karakterer=0;
        for(int i=0;i<tabel.length;i++)
          if(tabel[i] instanceof Karakter)
            karakterer += ((Karakter) tabel[i]).karakter;
        System.out.println("gennemsnit "+(karakterer*1.0/antal));
```

Spørgsmål 6.7

Opgaven er her løst med en vektor. Det kan også gøres med en tabel eller hægtet liste.

```
class Eksamensbevis extends Bevis{
  Vector v = new Vector();
  //Eksamensbevis(){ }
  public void tilfoej(Resultat k){ v.add(k); }
}
```

Spørgsmål 6.8

```
  public boolean studentOK(){
    int k,k1,k2,sum=0,antal=0;
    k1=13; k2=13; Object o;
    for(int i=0;i<v.size();i++){
      o = v.elementAt(i);
      if(o!=null&&o instanceof Karakter){
        k = ((Karakter) o).karakter;
        if (k<k1) {k2=k1; k1=k;}
        else if (k<k2) k2=k;
        antal++;
        sum +=k;
      }
    }
    return antal>=3&&(k1+k2+(sum-k1-k2)/(antal-2.0)>=13.0);
}
```