Seksjon 1

Oppgave	Tittel	Maks poeng	Oppgavetype
i	Innledning		Dokument
1	Oppgave 1: O-notasjon	8	Langsvar
2	Oppgave 2a: Binære søketrær	5	Programmering
3	Oppgave 2b: Binære søketrær	5	Programmering
4	Oppgave 2c: Binære søketrær	4	Muntlig
5	Oppgave 3a: Huffman	8	Muntlig
6	Oppgave 3b: Huffman	5	Muntlig
7	Oppgave 4a: CIA operasjon	4	Langsvar
8	Oppgave 4b: CIA operasjon	9	Programmering
9	Oppgave 4c: CIA operasjon	5	Langsvar
10	Oppgave 4d: CIA operasjon	4	Langsvar
11	Oppgave 5: Sekvenser	15	Programmering
12	Oppgave 6a: Sortering	13	Programmering
13	Oppgave 6b: Sortering	5	Langsvar
14	Oppgave 7a: Søking	6	Programmering
15	Oppgave 7b: Søking	4	Programmering

Innledning

INF2220 - Algoritmer og datastrukturer

Mandag 4. desember
Kl. 14.30 - 18.30 (4 timer)

Oppgavesettet består av totalt 15 (del)oppgaver.
Poengsum er angitt for hver oppgave.
Maksimum poengsum for hele oppgavesettet er 100 poeng.

Tillatte hjelpemidler: Alle trykte og skrevne.

I dette oppgavesettet skal du svare med digital håndtegning på oppgave 2c, 3a og 3b. Du bruker skisseark du får utdelt. Det er anledning til å bruke flere ark per oppgave. Se instruksjon for utfylling av skisseark på pult.

Det er IKKE anledning til å bruke digital håndtegning på andre oppgaver enn oppgave 2c, 3a og 3b. Det blir IKKE gitt ekstratid for å fylle ut informasjonsboksene på skisseark (engangskoder, kand.nr. o.l.).

Generelle råd:

- Husk å **begrunne** svar der det er bedt om.
- Skriv **korte** og **presise** kommentarer. Hvis du bruker kjente datastrukturer (list, set, map) trenger du ikke forklare hvordan de fungerer. Generelt: Hvis du bruker abstrakte datatyper fra biblioteket kan du bruke disse uten å forklare hva de gjør.
- **Vekten** till en oppgave indikerer **vanskelighetsgraden** og tidsbruken du bør bruke på oppgaven, ut i fra våre estimat. Dette kan være greit å benytte for å disponere tiden best mulig, dvs. ikke bruk mye tid på en oppgave med liten prosentsats (gå videre).
- **Står det at du skal skrive Java-kode**, så får du ikke poeng for pseudo-kode. Små kodefeil som manglende semikolon eller tilsvarende trekkes det ikke for.

Lykke til!

1 Oppgave 1: O-notasjon

Gitt følgende tre kodebiter:

```
// kodebit (i)
public void kodebit-i(int n) {
   int sum = 0;
   int x = 100;
   for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= x; j++) {
         sum += i*j;
      }
   }
}
// kodebit (ii)
public void kodebit-ii(int n) {
   int sum = 0;
   int x = 100;
   for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= n; j++) {
         sum += x;
      }
   }
}
// kodebit (iii)
public void kodebit-iii(int n) {
   int sum = 0;
   int x = 100;
   for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= n; j = j*2) {
         sum = n*x;
      }
   }
}
```

Svar på følgende tre spørsmål:

1) For hver av kodebitene over, hva er «worst-case» tidskompleksiteten, som funksjon av input-parameteren n?

2) Ut fra «worst-case»-analysen, hvilke(n) av kodebitene er mest effektiv?

3) Finnes det tilfeller hvor en av de andre kodebitene vil være mer effektiv? Begrunn svaret.

Skriv ditt svar her...

Maks poeng: 8

2

Oppgave 2a: Binære søketrær

Gitt følgende node-klasse:

```
class BinNode {
   int element;
   BinNode venstre;
   BinNode hoyre;
}
```

Skriv en metode

boolean sjekkStruktur(BinNode r, BinNode s) { … }

som sjekker om de to binær-trærne med røtter r og s er strukturelt like, det vil si at de har samme node-struktur når vi ser bort fra verdiene (elementene) i nodene. Lag eventuelt hjelpemetoder hvis det er hensiktsmessig.

Skriv ditt svar her...

Maks poeng: 5

3

Oppgave 2b: Binære søketrær

Gitt følgende node-klasse (samme som i oppgave 2a):

```
class BinNode {
   int element;
   BinNode venstre;
   BinNode hoyre;
}
```

En type balanserte søketrær er AVL-trær. Tilleggskravet er da at for hver node skal høyden til de to subtrærne ha en forskjell på max 1. Et tomt tre er definert til å ha høyde -1. Skriv en metode

boolean sjekkAVL(BinNode n) { … }

som sjekker om dette AVL-kravet er oppfylt for treet med rot-node n. Lag evt hjelpemetoder hvis det er hensiktsmessig.

Skriv ditt svar her...

Maks poeng: 5

4

Oppgave 2c: Binære søketrær

Fra oppgave 2b: En type balanserte søketrær er AVL-trær. Tilleggskravet er da at for hver node skal høyden til de to subtrærne ha en forskjell på max 1. Et tomt tre er definert til å ha høyde -1.

Ethvert AVL-tre kan farges som et rød-svart tre, men ikke alle rød-svarte trær oppfyller AVL-kravet. Tegn et eksempel på et rød-svart tre (med korrekt fargekoding) som IKKE tilfredsstiller AVL-kravet. Angi fargen til hver node tydelig med R (rød) eller S (svart).

I denne oppgaven skal du svare med digital håndtegning. Bruk eget skisseark (utdelt). Se instruksjon for utfylling av skisseark på pult.

Maks poeng: 4

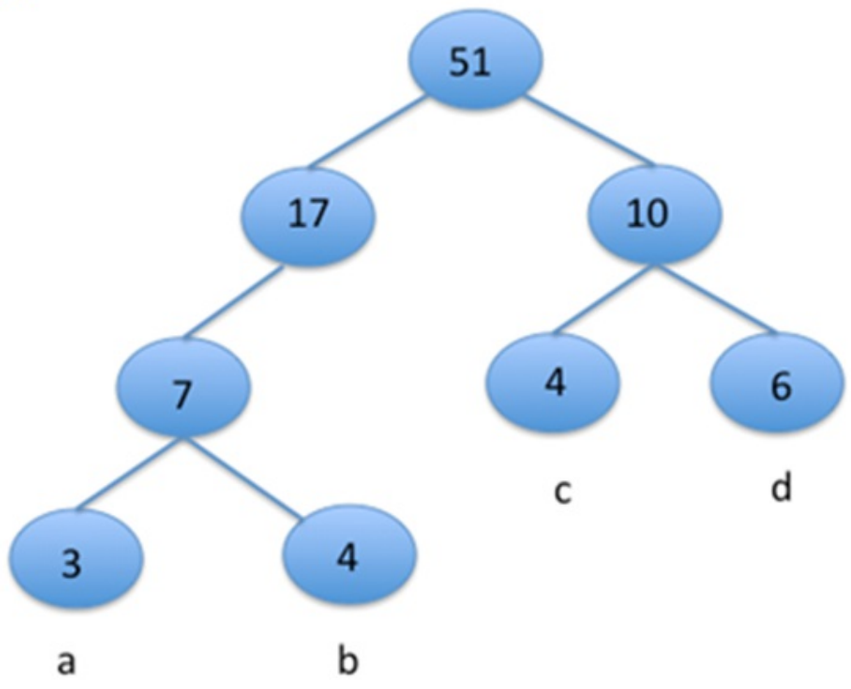
5

Oppgave 3a: Huffman

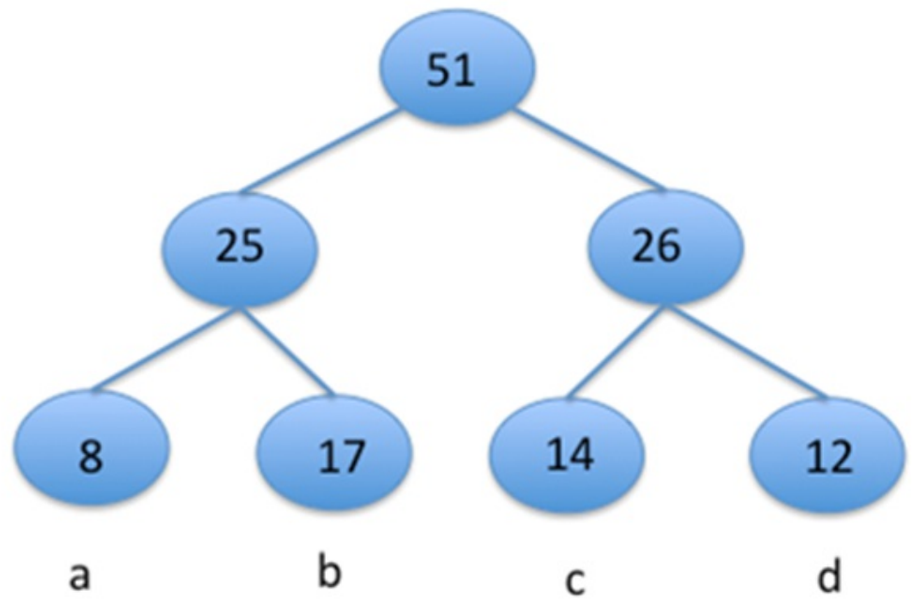
Gitt de to trærne A og B under. For hvert tre, er treet et gyldig Huffman-tre? Hvis det ikke er gyldig, forklar hvorfor og endre det slik at det blir gyldig. Du kan anta at vektingen til løvnodene er riktig.

I denne oppgaven skal du svare med digital håndtegning. Bruk eget skisseark (utdelt). Se instruksjon for utfylling av skisseark på pult.

A:



B:



Maks poeng: 8

6

Oppgave 3b: Huffman

Gitt tekststrengen “AADDCCCEEEEE”, vis Huffman-treet og -koden for hvert tegn i strengen.

I denne oppgaven skal du svare med digital håndtegning. Bruk eget skisseark (utdelt). Se instruksjon for utfylling av skisseark på pult.

Maks poeng: 5

7

Oppgave 4a: CIA operasjon

Den samlede teksten for oppgave 4 finnes i vedlagte pdf-dokument. Av hensyn til sensuren er det viktig at du svarer på hver deloppgave for seg, der det er satt av plass til dette.

Oppgave 4a

Svar på oppgave 4a i vedlegget.

Skriv ditt svar her...

Maks poeng: 4

8

Oppgave 4b: CIA operasjon

Den samlede teksten for oppgave 4 finnes i vedlagte pdf-dokument. Av hensyn til sensuren er det viktig at du svarer på hver deloppgave for seg, der det er satt av plass til dette.

Oppgave 4b

Svar på oppgave 4b i vedlegget.

Skriv ditt svar her...

Maks poeng: 9

9

Oppgave 4c: CIA operasjon

Den samlede teksten for oppgave 4 finnes i vedlagte pdf-dokument. Av hensyn til sensuren er det viktig at du svarer på hver deloppgave for seg, der det er satt av plass til dette.

Oppgave 4c

Svar på oppgave 4c i vedlegget.

Skriv ditt svar her...

Maks poeng: 5

10

Oppgave 4d: CIA operasjon

Den samlede teksten for oppgave 4 finnes i vedlagte pdf-dokument. Av hensyn til sensuren er det viktig at du svarer på hver deloppgave for seg, der det er satt av plass til dette.

Oppgave 4d

Svar på oppgave 4d i vedlegget.

Skriv ditt svar her...

Maks poeng: 4

11

Oppgave 5: Sekvenser

Ta utgangspunkt i de n heltallene fra 0 til n-1. Skriv et komplett program som for en gitt n genererer alle sekvenser av lengde 2n slik at

- alle tallene fra 0 til n-1 brukes nøyaktig to ganger.
- like tall IKKE står rett ved siden av hverandre i sekvensen.

Skriv ditt svar her...

Maks poeng: 15

12

Oppgave 6a: Sortering

Den samlede teksten for oppgave 6 finnes i vedlagte pdf-dokument. Av hensyn til sensuren er det viktig at du svarer på hver deloppgave for seg, der det er satt av plass til dette.

Oppgave 6a

Skriv metoden insertABSort som spesifisert i vedlegget.

Skriv ditt svar her...

Maks poeng: 13

13

Oppgave 6b: Sortering

Den samlede teksten for oppgave 6 finnes i vedlagte pdf-dokument. Av hensyn til sensuren er det viktig at du svarer på hver deloppgave for seg, der det er satt av plass til dette.

Oppgave 6b

Svar på oppgave 6b i vedlegget.

Skriv ditt svar her...

Maks poeng: 5

14

Oppgave 7a: Søking

Felles introduksjon til denne og neste deloppgave:

Disse to deloppgavene handler om å skrive Java-koder for metoder som finner medianen av tallene i arrayen int[] a. Definisjonen av medianen er at det er like mange elementer i a[] som er mindre og like mange som er større enn den. I begge deloppgavene skal du løse dette **uten å sortere** arrayen a[] først. Det betyr at du må gå gjennom elementene for å sjekke om de tilfredsstiller kravene til en median. Du får altså ikke noen poeng for først å sortere a[] og så ta ut midt-elementet som svar.

Oppgave 7a

For enkelthets skyld kan du her anta at alle elementene i a[] har forskjellig verdi og at det er et odde antall elementer i a[]. Du skal skrive Javakode for en metode

int median1(int[] a)

som da finner medianen**.** Forklar deretter i en kommentar nederst hva kompleksiteten til metoden din er (i O-notasjon), og sammenlign effektiviteten med det å først sortere og så returnere midtelementet – er sortering først alltid mest effektivt?

Skriv ditt svar her...

Maks poeng: 6

15

Oppgave 7b: Søking

Felles introduksjon til denne og forrige deloppgave:

Disse to deloppgavene handler om å skrive Java-koder for metoder som finner medianen av tallene i arrayen int[] a. Definisjonen av medianen er at det er like mange elementer i a[] som er mindre og like mange som er større enn den. I begge deloppgavene skal du løse dette **uten å sortere** arrayen a[] først. Det betyr at du må gå gjennom elementene for å sjekke om de tilfredsstiller kravene til en median. Du får altså ikke noen poeng for først å sortere a[] og så ta ut midt-elementet som svar.

Oppgave 7b

Vi fjerner nå antagelsen om at alle elementene i a[] har forskjellig verdi. Skriv Javakode for en metode

int median2(int[] a)

som da finner medianverdien i a[]. Gi også her hvilken kompleksitet denne algoritmen har.

Skriv ditt svar her...

Maks poeng: 4

Question 10
Attached

Oppgave 4: CIA-operasjon (totalt 22 poeng)

Du er en CIA-agent som fungerer som informasjonsskilde til N spioner. Hver spion kan utveksle informasjon med utvalgte andre spioner over sikre eller usikre kanaler. En kanal går mellom nøyaktig to spioner og hver kanal kan kommunisere begge veier. Når informasjonen du skal dele er topphemmelig, kan spionene kun bruke sikre kanaler for å spre informasjon de har fått fra deg videre. Gitt spionene og deres kanaler (hvem de er i mellom og om de er sikre), lag en effektiv algoritme som beregner det minimale antallet spioner som du må fortelle hemmeligheten direkte til for at de senere skal kunne distribuere informasjonen til *alle* spionene i nettverket ved å kun bruke de sikre kanalene. Anta at du som CIA-agent har en liste over alle spionene og informasjon om kanalene:

```
class Kanal {
    Spion a;
    Spion b;
    boolean sikker; // true hvis kanalen er sikker

    Kanal(Spion a, Spion b, boolean sikker) {
        this.a = a;
        this.b = b;
        this.sikker = sikker;
    }

    Spion destinasjon(Spion kilde) {
        return (kilde == a) ? b : (kilde == b) ? a : null;
    }
}


class Spion {
    List <Kanal> kanaler;
    ...
}
```

Deloppgave 4a – (4 poeng)

Forklar kort hvilken datastruktur og algoritme du kan bruke for å representere og løse dette problemet.

Deloppgave 4b – (9 poeng)

Implementer algoritmen med signaturen

```
int antallSpioner(List <Spion> spioner)
```

Deloppgave 4c – (5 poeng)

Det er nye tider og du har ikke tid til å dele informasjon direkte med så mange spioner. Du vil fortelle informasjonen til så få som overhodet mulig, så du aksepterer at de bruker usikker kommunikasjon dersom det ikke finnes et sikkert alternativ. Du ønsker altså helst at de skal benytte sikre kanaler for å spre informasjon så langt det lar seg gjøre, men de kan i tillegg også dele informasjon

over usikre kanaler hvis nødvendig. Hvilken effektiv algoritme kan du bruke for å finne det minimale antallet usikre kanaler som må benyttes dersom informasjonen skal nå alle spionene? Du må også kunne identifisere disse usikre kanalene. Forklar hvordan du kan bruke denne algoritmen.

Deloppgave 4d – (4 poeng)

For 8 spioner med kanalene:

Kanal(A, B, true)
Kanal(A, D, false)
Kanal(B, D, true)
Kanal(B,C, false)
Kanal(C, F, true)
Kanal(D,F, false)
Kanal(C, E, false)
Kanal(G,H, false)

Basert på deloppgave 4c), hvor mange usikre kanaler må det benyttes og hvorfor? Identifiser disse kanalene.

Question 12
Attached

Oppgave 6 – Sortering (totalt 18 poeng)

Deloppgave 6a – (13 poeng)

Nederst i denne oppgaven er gitt Javakode for insertSort av et område i en array a[left..right] - dvs. sorterer a[] stigende fra og med element a[left] til og med element a[right].
I denne oppgaven skal du lage Java-kode for følgende metode som leser elementene i en usortert array a[left..right] og flytter dem over i b[left..right] slik at b[left..right] da blir sortert:

```
insertABSort (int[]a, int b[], int left, int right) {...}
```

som skal tilfredstille følgende tre krav:

1) a[left..right] er usortert, og skal ikke sorteres i metoden og følgelig være uendret når metoden er ferdig.
2) b[left..right] skal etter denne overflyttingen inneholde elementene fra a[left..right] sortert i stigende rekkefølge.
3) Det er ikke lov å først kopiere over a[left..right] til b[left..right] og så sortere b[left..right], Selve sorteringen av b[] skal skje element for element når de flyttes over (bla. fordi dette er raskest)

Ta gjerne utgangspunkt i følgende kode og gjør de tillegg/fratrekk og endringer som løser oppgaven:

```
// Innstikksortering av a[left..right]
static void insertSort(int a[],int left, int right){
    if ( right-left > 0) {
        int i, t;

        for (int k = left ; k < right ; k++){
            t = a[k+1];
            i = k;
            while (i >= left && a[i] > t) {
                a[i+1] = a[i];
                i--;
            }
            a[i+1] = t;
        } // end for k
    } // end len > 1
} // end insertSort
```

Deloppgave 6b – (5 poeng)
Hva blir resultatet av metoden din hvis den kalles med metodekallet
`insertABSort` (a,a,left,right)
dvs. hvis du bruker arrayen a[] på begge parameter-plassene, vil dette bli vanlig innstikksortering av a[left..right] eller ikke? Begrunn svaret.

Question 13
Attached

Oppgave 6 – Sortering (totalt 18 poeng)

Deloppgave 6a – (13 poeng)

Nederst i denne oppgaven er gitt Javakode for insertSort av et område i en array a[left..right] - dvs. sorterer a[] stigende fra og med element a[left] til og med element a[right].
I denne oppgaven skal du lage Java-kode for følgende metode som leser elementene i en usortert array a[left..right] og flytter dem over i b[left..right] slik at b[left..right] da blir sortert:

```
insertABSort (int[]a, int b[], int left, int right) {...}
```

som skal tilfredstille følgende tre krav:

1) a[left..right] er usortert, og skal ikke sorteres i metoden og følgelig være uendret når metoden er ferdig.
2) b[left..right] skal etter denne overflyttingen inneholde elementene fra a[left..right] sortert i stigende rekkefølge.
3) Det er ikke lov å først kopiere over a[left..right] til b[left..right] og så sortere b[left..right], Selve sorteringen av b[] skal skje element for element når de flyttes over (bla. fordi dette er raskest)

Ta gjerne utgangspunkt i følgende kode og gjør de tillegg/fratrekk og endringer som løser oppgaven:

```
// Innstikksortering av a[left..right]
static void insertSort(int a[],int left, int right){
    if ( right-left > 0) {
        int i, t;

        for (int k = left ; k < right ; k++){
            t = a[k+1];
            i = k;
            while (i >= left && a[i] > t) {
                a[i+1] = a[i];
                i--;
            }
            a[i+1] = t;
        } // end for k
    } // end len > 1
} // end insertSort
```

Deloppgave 6b – (5 poeng)
Hva blir resultatet av metoden din hvis den kalles med metodekallet
`insertABSort` (a,a,left,right)
dvs. hvis du bruker arrayen a[] på begge parameter-plassene, vil dette bli vanlig innstikksortering av a[left..right] eller ikke? Begrunn svaret.

Question 7
Attached

Oppgave 4: CIA-operasjon (totalt 22 poeng)

Du er en CIA-agent som fungerer som informasjonsskilde til N spioner. Hver spion kan utveksle informasjon med utvalgte andre spioner over sikre eller usikre kanaler. En kanal går mellom nøyaktig to spioner og hver kanal kan kommunisere begge veier. Når informasjonen du skal dele er topphemmelig, kan spionene kun bruke sikre kanaler for å spre informasjon de har fått fra deg videre. Gitt spionene og deres kanaler (hvem de er i mellom og om de er sikre), lag en effektiv algoritme som beregner det minimale antallet spioner som du må fortelle hemmeligheten direkte til for at de senere skal kunne distribuere informasjonen til *alle* spionene i nettverket ved å kun bruke de sikre kanalene. Anta at du som CIA-agent har en liste over alle spionene og informasjon om kanalene:

```
class Kanal {
    Spion a;
    Spion b;
    boolean sikker; // true hvis kanalen er sikker

    Kanal(Spion a, Spion b, boolean sikker) {
        this.a = a;
        this.b = b;
        this.sikker = sikker;
    }

    Spion destinasjon(Spion kilde) {
        return (kilde == a) ? b : (kilde == b) ? a : null;
    }
}


class Spion {
    List <Kanal> kanaler;
    ...
}
```

Deloppgave 4a – (4 poeng)

Forklar kort hvilken datastruktur og algoritme du kan bruke for å representere og løse dette problemet.

Deloppgave 4b – (9 poeng)

Implementer algoritmen med signaturen

```
int antallSpioner(List <Spion> spioner)
```

Deloppgave 4c – (5 poeng)

Det er nye tider og du har ikke tid til å dele informasjon direkte med så mange spioner. Du vil fortelle informasjonen til så få som overhodet mulig, så du aksepterer at de bruker usikker kommunikasjon dersom det ikke finnes et sikkert alternativ. Du ønsker altså helst at de skal benytte sikre kanaler for å spre informasjon så langt det lar seg gjøre, men de kan i tillegg også dele informasjon

over usikre kanaler hvis nødvendig. Hvilken effektiv algoritme kan du bruke for å finne det minimale antallet usikre kanaler som må benyttes dersom informasjonen skal nå alle spionene? Du må også kunne identifisere disse usikre kanalene. Forklar hvordan du kan bruke denne algoritmen.

Deloppgave 4d – (4 poeng)

For 8 spioner med kanalene:

Kanal(A, B, true)
Kanal(A, D, false)
Kanal(B, D, true)
Kanal(B,C, false)
Kanal(C, F, true)
Kanal(D,F, false)
Kanal(C, E, false)
Kanal(G,H, false)

Basert på deloppgave 4c), hvor mange usikre kanaler må det benyttes og hvorfor? Identifiser disse kanalene.

Question 8
Attached

Oppgave 4: CIA-operasjon (totalt 22 poeng)

Du er en CIA-agent som fungerer som informasjonsskilde til N spioner. Hver spion kan utveksle informasjon med utvalgte andre spioner over sikre eller usikre kanaler. En kanal går mellom nøyaktig to spioner og hver kanal kan kommunisere begge veier. Når informasjonen du skal dele er topphemmelig, kan spionene kun bruke sikre kanaler for å spre informasjon de har fått fra deg videre. Gitt spionene og deres kanaler (hvem de er i mellom og om de er sikre), lag en effektiv algoritme som beregner det minimale antallet spioner som du må fortelle hemmeligheten direkte til for at de senere skal kunne distribuere informasjonen til *alle* spionene i nettverket ved å kun bruke de sikre kanalene. Anta at du som CIA-agent har en liste over alle spionene og informasjon om kanalene:

```
class Kanal {
    Spion a;
    Spion b;
    boolean sikker; // true hvis kanalen er sikker

    Kanal(Spion a, Spion b, boolean sikker) {
        this.a = a;
        this.b = b;
        this.sikker = sikker;
    }

    Spion destinasjon(Spion kilde) {
        return (kilde == a) ? b : (kilde == b) ? a : null;
    }
}


class Spion {
    List <Kanal> kanaler;
    ...
}
```

Deloppgave 4a – (4 poeng)

Forklar kort hvilken datastruktur og algoritme du kan bruke for å representere og løse dette problemet.

Deloppgave 4b – (9 poeng)

Implementer algoritmen med signaturen

```
int antallSpioner(List <Spion> spioner)
```

Deloppgave 4c – (5 poeng)

Det er nye tider og du har ikke tid til å dele informasjon direkte med så mange spioner. Du vil fortelle informasjonen til så få som overhodet mulig, så du aksepterer at de bruker usikker kommunikasjon dersom det ikke finnes et sikkert alternativ. Du ønsker altså helst at de skal benytte sikre kanaler for å spre informasjon så langt det lar seg gjøre, men de kan i tillegg også dele informasjon

over usikre kanaler hvis nødvendig. Hvilken effektiv algoritme kan du bruke for å finne det minimale antallet usikre kanaler som må benyttes dersom informasjonen skal nå alle spionene? Du må også kunne identifisere disse usikre kanalene. Forklar hvordan du kan bruke denne algoritmen.

Deloppgave 4d – (4 poeng)

For 8 spioner med kanalene:

Kanal(A, B, true)
Kanal(A, D, false)
Kanal(B, D, true)
Kanal(B,C, false)
Kanal(C, F, true)
Kanal(D,F, false)
Kanal(C, E, false)
Kanal(G,H, false)

Basert på deloppgave 4c), hvor mange usikre kanaler må det benyttes og hvorfor? Identifiser disse kanalene.

Question 9
Attached

Oppgave 4: CIA-operasjon (totalt 22 poeng)

Du er en CIA-agent som fungerer som informasjonsskilde til N spioner. Hver spion kan utveksle informasjon med utvalgte andre spioner over sikre eller usikre kanaler. En kanal går mellom nøyaktig to spioner og hver kanal kan kommunisere begge veier. Når informasjonen du skal dele er topphemmelig, kan spionene kun bruke sikre kanaler for å spre informasjon de har fått fra deg videre. Gitt spionene og deres kanaler (hvem de er i mellom og om de er sikre), lag en effektiv algoritme som beregner det minimale antallet spioner som du må fortelle hemmeligheten direkte til for at de senere skal kunne distribuere informasjonen til *alle* spionene i nettverket ved å kun bruke de sikre kanalene. Anta at du som CIA-agent har en liste over alle spionene og informasjon om kanalene:

```
class Kanal {
    Spion a;
    Spion b;
    boolean sikker; // true hvis kanalen er sikker

    Kanal(Spion a, Spion b, boolean sikker) {
        this.a = a;
        this.b = b;
        this.sikker = sikker;
    }

    Spion destinasjon(Spion kilde) {
        return (kilde == a) ? b : (kilde == b) ? a : null;
    }
}


class Spion {
    List <Kanal> kanaler;
    ...
}
```

Deloppgave 4a – (4 poeng)

Forklar kort hvilken datastruktur og algoritme du kan bruke for å representere og løse dette problemet.

Deloppgave 4b – (9 poeng)

Implementer algoritmen med signaturen

```
int antallSpioner(List <Spion> spioner)
```

Deloppgave 4c – (5 poeng)

Det er nye tider og du har ikke tid til å dele informasjon direkte med så mange spioner. Du vil fortelle informasjonen til så få som overhodet mulig, så du aksepterer at de bruker usikker kommunikasjon dersom det ikke finnes et sikkert alternativ. Du ønsker altså helst at de skal benytte sikre kanaler for å spre informasjon så langt det lar seg gjøre, men de kan i tillegg også dele informasjon

over usikre kanaler hvis nødvendig. Hvilken effektiv algoritme kan du bruke for å finne det minimale antallet usikre kanaler som må benyttes dersom informasjonen skal nå alle spionene? Du må også kunne identifisere disse usikre kanalene. Forklar hvordan du kan bruke denne algoritmen.

Deloppgave 4d – (4 poeng)

For 8 spioner med kanalene:

Kanal(A, B, true)
Kanal(A, D, false)
Kanal(B, D, true)
Kanal(B,C, false)
Kanal(C, F, true)
Kanal(D,F, false)
Kanal(C, E, false)
Kanal(G,H, false)

Basert på deloppgave 4c), hvor mange usikre kanaler må det benyttes og hvorfor? Identifiser disse kanalene.