Høgskolen i Sørøst-Norge

EKSAMENSFORSIDE

Skriftlig eksamen med tilsyn

Emnekode: 5610	Emnenavn: Algoritmer og datastrukturer	
Dato: 22.05.2017	Tid fra/til: 09.00 – 14.00	Antall timer: 5
Ansvarlig faglærer: Tor Lønnestad		
Campus: Bø	Fakultet: Handelshøgskolen	
Antall oppgaver: 3 (15 deloppgaver)	Antall vedlegg: 4	Ant. sider inkl. forside og vedlegg:15
Tillatte hjelpemidler (jfr. emnebeskrivelse): Alle trykte og skrevne		
Opplysninger om vedlegg: Vedlegg A: Dokumentasjon av ArrayList - utdrag Vedlegg B: class Edge, class Vertex og class Graph Vedlegg C: class GTree og class GNode Vedlegg D: Variabelt antall parametre		
Merknader:		

Kryss av for type eksamenspapir	Ruter	☐	Linjer	☐

KANDIDATEN MÅ SELV KONTROLLERE AT OPPGAVESETTET ER FULLSTENDIG

Råd og retningslinjer. Les oppgaveteksten godt før du går i gang med å løse oppgaven. Deloppgavene er uavhengige av hverandre i den forstand at om du ikke får til en oppgave, kan du likevel gjøre neste, som om den første var løst. Fordél tiden godt på alle oppgavene. Om du mener en oppgave er upresis, så skriv din egen presisering. Lag de hjelpemetoder du måtte ønske.

Oppgave 1 – ArrayList (30%)

Du liker ArrayList-klassen, men synes det mangler noen metoder. Du skal derfor lage en subklasse av ArrayList. Se vedlegg A for utdrag av dokumentasjon av ArrayList.

1a (7%)

Lag klassen MyArrayList som subklasse av ArrayList. Den skal være generisk (som ArrayList) og ha metoden

```
public void mirror(){ ... }
```

som speiler innholdet, dvs. snur rekkefølgen på elementene. Hva blir algoritmens orden?

1b (7%)

Lag metoden

```
public boolean sorted ( ... ){ ... }
```

i klassen MyArrayList. Den skal ha en Comparator-parameter. Ved hjelp av denne skal metoden undersøke om tabellen er sortert, og returnere true dersom det er tilfelle, og false ellers. Hva blir algoritmens orden?

1c (7%)

Lag metoden

```
public void duplicate( ... ){ ... }
```

i klassen MyArrayList. Metoden skal ha en heltallsparameter som styrer hvor mange referanser som skal opprettes til hvert element i tabellen. Dersom tabellen inneholder ["Per", "Kari", "Ola"] og kallet duplicate(2) utføres, skal tabellen etterpå inneholde ["Per", "Per", "Kari", "Kari", "Ola", "Ola"]. Hva blir algoritmens orden?

1d (9%)

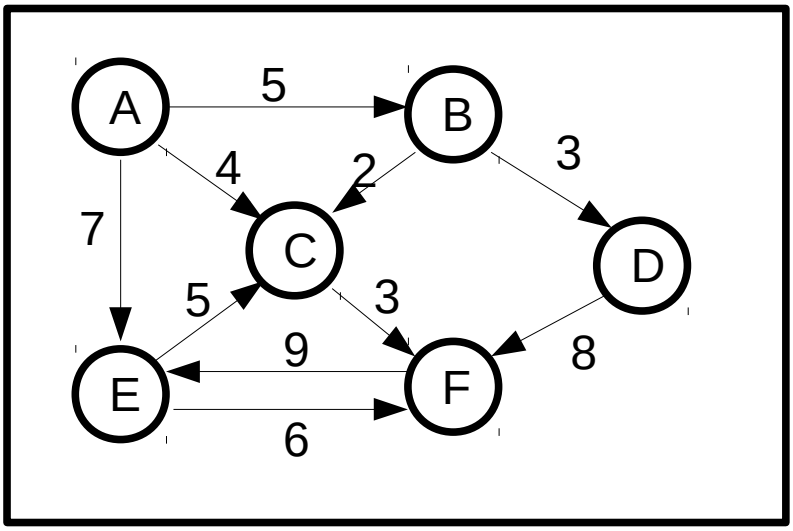
Lag metoden

```
public void select( ... ){ ... }
```

i klassen MyArrayList. Den skal bare beholde noen av referansene i tabellen, en heltallsparameter styrer utplukket. Første element (element 0) skal alltid beholdes, og deretter beholdes eksempelvis hvert tredje element dersom parameteren er 3. Alle andre referanser skal slettes. Dersom tabellen inneholder ["Per", "Kari", "Ola", "Åse", "Jens", "Gro", "Liv"] og kallet select(3) utføres, skal tabellen etterpå inneholde ["Per", "Åse", "Liv"]. Hva blir algoritmens orden?

Oppgave 2 – Graf (35%)

Figur 1 viser en rettet, vektet graf.



Figur 1 – En rettet, vektet graf

2a (5%)

- Vis hvordan en tekstfil som beskriver denne grafen, typisk ser ut.
- Forklar filformatet.

2b (5%)

Vis naboskapsmatrise-representasjon av denne grafen. Tegn matrisen så godt det lar seg gjøre i eksamensverktøyet.

2c (15%)

Grafen i figur 1 inneholder som du ser sykler (cycles). Du skal her lage en metode for å rapportere om en rettet graf har sykler. En enkel algoritme for å avgjøre dette har følgende pseudokode:

```
for (alle noder)
   if (det er mulig å gå fra noden rekursivt til alle naboer og komme tilbake til noden)
      return true;
return false;
```

Koden skal baseres på klassene Edge, Vertex og Graph i læreboka, se vedlegg B.
For å oppdage at man har kommet tilbake til noden man startet fra, kan man for eksempel merke startnoden ved hjelp av scratch-variablen. Dersom den rekursive algoritmen har gått flere rekursive steg enn det er noder i grafen uten å komme til startnoden, kan du returnere false. Det finnes da rett nok en sykel, men den omfatter ikke noden man startet fra.

Du skal altså lage metoden
`public boolean hasCycles() { ... }`
i class Graph. Det anbefales å lage denne ikke-rekursiv, og lage en rekursiv hjelpemetode.

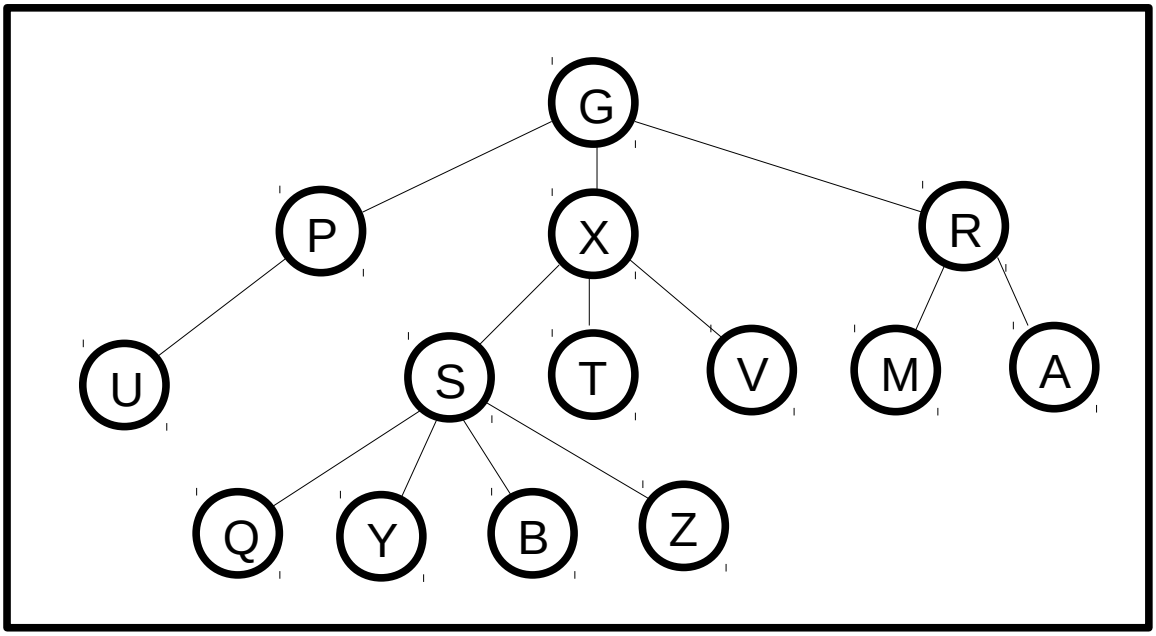
2d (5%)

Lag en main-metode som oppretter et objekt av klassen Graph, bygger grafen i figur 1, og rapporterer til System.out om den har sykler eller ikke – ved å kalle metoden i 2c.

2e (5%)
Hvordan skal koden modifiseres dersom du ønsker å få rapportert hvilke noder som inngår i den sykelen som eventuelt oppdages? Vis kode, eller pseudokode, eller forklar med ord. Kode gir best uttelling.

Oppgave 3 Generelle trestrukturer (35%)

Figur 2 viser et generelt tre.



Figur 2 - Generelt tre

3a (5%)
Skriv elementene i treet i figur 2 i preorden, postorden og nivåorden rekkefølge.

3b (6%)
class GTree og class GNode er skissert i vedlegg C.

- Lag konstruktørmetode i class GNode som tar én parameter: et element.
- Lag konstruktørmetode i class GTree som tar som parametre et element og et variabelt antall eksisterende trær. Elementet skal legges i rotnoden av det nye treet, og rotnodene i de eksisterende trærne skal bli barn av rotnoden i det nye. Se vedlegg D angående variabelt antall parametre.

3c (6%)
Lag metodene
```
public void printPreorder() { ... }
public void printPostorder() { ... }
```
i class GTree, slik at de skriver ut alle elementene i treet i preorden/postorden rekkefølge. De skal lages slik at de kaller rekursive instansmetoder i class GNode, disse skal du også lage.

3d (6%)
Lag metoden
```
public void printLevelorder() { ... }
```
i class GTree slik at den skriver ut alle elementene i treet i nivåorden rekkefølge.

3e (6%)
Skriv metoden
```
public int size() { ... }
```
i class GTree slik at den returnerer antall elementer i treet. Du velger selv om du vil lage hjelpemetoder og i hvilken klasse.

3f (6%)
Skriv en main-metode som bygger opp treet vist i figur 2 og deretter kaller de tre utskriftsmetodene.

Lykke til!

Råd og retningsliner. Les oppgåveteksten godt før du går i gang med å løyse oppgåva. Deloppgåvene er uavhengige av kvarandre i den meining at om du ikkje får til ei oppgåve, kan du likevel gjere neste, som om den fyrste var løyst. Fordél tida godt på alle oppgåvene. Om du meiner ei oppgåve er upresis, så skriv din eigen presisering. Lag dei hjelpemetodar du måtte ynskje.

Oppgåve 1 – ArrayList (30%)

Du likar ArrayList-klassa, men synast det manglar nokre metodar. Du skal difor lage ei subklasse av ArrayList. Sjå vedlegg A for utdrag av dokumentasjon av ArrayList.

1a (7%)

Lag klassa MyArrayList som subklasse av ArrayList. Den skal vera generisk (som ArrayList) og ha metoden

```
public void mirror(){ ... }
```

som speglar innhaldet, dvs. snur rekkefylgja på elementa. Kva vert algoritma si orden?

1b (7%)

Lag metoden

```
public boolean sorted ( ... ){ ... }
```

i klassa MyArrayList. Den skal ha ein Comparator-parameter. Ved hjelp av denne skal metoden undersøke om tabellen er sortert, og returnere true dersom det er tilfelle, og false elles. Kva vert algoritma si orden?

1c (7%)

Lag metoden

```
public void duplicate( ... ){ ... }
```

i klassa MyArrayList. Metoden skal ha ein heiltalsparameter som styrer kor mange referansar som skal opprettast til kvart element i tabellen. Dersom tabellen inneheld ["Per", "Kari", "Ola"] og kallet duplicate(2) vert utført, skal tabellen etterpå innehalde ["Per", "Per", "Kari", "Kari", "Ola", "Ola"]. Kva vert algoritma si orden?

1d (9%)

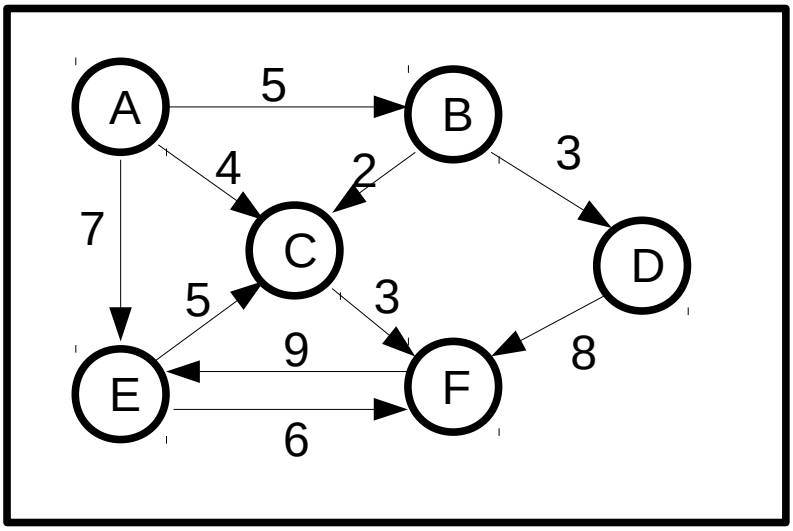
Lag metoden

```
public void select( ... ){ ... }
```

i klassa MyArrayList. Den skal berre behalde nokre av referansane i tabellen, ein heiltalsparameter styrer utplukket. Fyrste element (element 0) skal alltid behaldast, og deretter behaldast til dømes kvart tredje element dersom parameteren er 3. Alle andre referansar skal slettast. Dersom tabellen inneheld ["Per", "Kari", "Ola", "Åse", "Jens", "Gro", "Liv"] og kallet select(3) vert utført, skal tabellen etterpå innehalde ["Per", "Åse", "Liv"]. Kva vert algoritma si orden?

Oppgåve 2 – Graf (35%)

Figur 1 viser ein retta, vekta graf.



Figur 1 – Ein retta, vekta graf

2a (5%)

- Vis korleis ei tekstfil som beskriver denne grafen, typisk ser ut.
- Forklar filformatet.

2b (5%)

Vis naboskapsmatrise-representasjon av denne grafen. Teikn matrisa så godt det let seg gjera i eksamensverktyet.

2c (15%)

Grafen i figur 1 inneheld som du ser syklar (cycles). Du skal her lage ein metode for å rapportere om ein retta graf har syklar. Ei enkel algoritme for å avgjera dette har fylgjande pseudokode:

```
for (alle nodar)
   if (det er mogeleg å gå frå noden rekursivt til alle naboar og kome tilbake til noden)
      return true;
return false;
```

Koden skal baserast på klassene Edge, Vertex og Graph i læreboka, sjå vedlegg B.
For å oppdage at ein har kome tilbake til noden ein starta frå, kan ein til dømes merke startnoden ved hjelp av scratch-variablen. Dersom den rekursive algoritma har gått fleire rekursive steg enn det er nodar i grafen utan å kome til startnoden, kan du returnere false. Det finst då rett nok ein sykel, men den omfattar ikkje noden ein starta frå.

Du skal altså lage metoden
`public boolean hasCycles() { ... }`
i class Graph. Det vert tilrådd å lage denne ikkje-rekursiv, og lage ei rekursiv hjelpemetode.

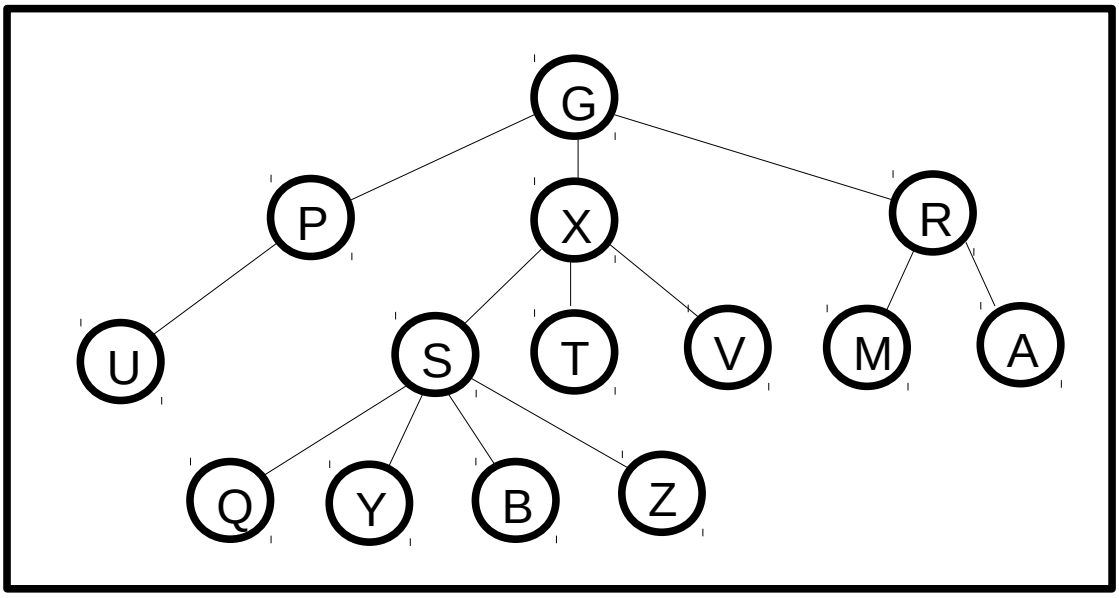
2d (5%)

Lag ein main-metode som opprettar eit objekt av klassa Graph, byggjer grafen i figur 1, og rapporterer til System.out om den har syklar eller ikkje – ved å kalle metoden i 2c.

2e (5%)
Korleis skal koden modifiserast dersom du ynskjer å få rapportert kva for nodar som inngår i den sykelen som eventuelt vert oppdaga? Vis kode, eller pseudokode, eller forklar med ord. Kode gjev best uttelling.

Oppgåve 3 Generelle trestrukturar (35%)

Figur 2 viser eit generelt tre.



Figur 2 - Generelt tre

3a (5%)
Skriv elementa i treet i figur 2 i preorden, postorden og nivåorden rekkefylgje.

3b (6%)
class GTree og class GNode er skissert i vedlegg C.

- Lag konstruktørmetode i class GNode som tek ein parameter: eit element.
- Lag konstruktørmetode i class GTree som tek som parametre eit element og eit variabelt antal eksisterande tre. Elementet skal leggast i rotnoden av det nye treet, og rotnodane i dei eksisterande trea skal bli barn av rotnoden i det nye. Sjå vedlegg D når det gjeld variabelt antal parametre.

3c (6%)
Lag metodane
```
public void printPreorder() { ... }
public void printPostorder() { ... }
```
i class GTree, slik at dei skriv ut alle elementa i treet i preorden/postorden rekkefylgje. Dei skal lagast slik at dei kallar rekursive instansmetodar i class GNode, desse skal du også lage.

3d (6%)
Lag metoden
```
public void printLevelorder() { ... }
```
i class GTree slik at den skriv ut alle elementa i treet i nivåorden rekkefylgje.

3e (6%)
Skriv metoden
`public int size() { ... }`
i class GTree slik at den returnerer antal element i treet. Du vel sjølv om du vil lage hjelpemetodar og i kva for klasse.

3f (6%)
Skriv ein main-metode som bygger opp treet vist i figur 2 og deretter kallar dei tre utskriftsmetodane.

Lykke til!

Vedlegg A: ArrayList Methods – utdrag

Modifier and Type	Method and Description
boolean	**add**(E e) Appends the specified element to the end of this list.
void	**add**(int index, E element) Inserts the specified element at the specified position in this list.
boolean	**addAll**(Collection<? extends E> c) Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator.
void	**clear**() Removes all of the elements from this list.
Object	**clone**() Returns a shallow copy of this ArrayList instance.
boolean	**contains**(Object o) Returns true if this list contains the specified element.
void	**ensureCapacity**(int minCapacity) Increases the capacity of this ArrayList instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.
E	**get**(int index) Returns the element at the specified position in this list.
int	**indexOf**(Object o) Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
boolean	**isEmpty**() Returns true if this list contains no elements.
Iterator<E>	**iterator**() Returns an iterator over the elements in this list in proper sequence.
int	**lastIndexOf**(Object o) Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
ListIterator<E>	**listIterator**() Returns a list iterator over the elements in this list (in proper sequence).
ListIterator<E>	**listIterator**(int index) Returns a list-iterator of the elements in this list (in proper sequence), starting at the specified position in the list.
E	**remove**(int index) Removes the element at the specified position in this list.
boolean	**remove**(Object o) Removes the first occurrence of the specified element from this list, if it is present.

`boolean`	**`removeAll`**`(Collection<?> c)` Removes from this list all of its elements that are contained in the specified collection.
`protected void`	**`removeRange`**`(int fromIndex, int toIndex)` Removes from this list all of the elements whose index is between `fromIndex`, inclusive, and `toIndex`, exclusive.
`boolean`	**`retainAll`**`(Collection<?> c)` Retains only the elements in this list that are contained in the specified collection.
`E`	**`set`**`(int index, E element)` Replaces the element at the specified position in this list with the specified element.
`int`	**`size`**`()` Returns the number of elements in this list.
`Object[]`	**`toArray`**`()` Returns an array containing all of the elements in this list in proper sequence (from first to last element).
`<T> T[]`	**`toArray`**`(T[] a)` Returns an array containing all of the elements in this list in proper sequence (from first to last element); the runtime type of the returned array is that of the specified array.

Metoder som er arvet fra superklasser:

`equals, hashCode, subList, containsAll, toString, finalize, getClass, notify, notifyAll, wait, subList`

Vedlegg B : class Edge, class Vertex og class Graph

```
class Edge {
    public Vertex dest;
    public double cost;

    public Edge(Vertex d, double c) {
        dest = d; cost = c;
    }
}

class Vertex
{
    public String name;
    public List<Edge> adj;
    public double dist;
    public Vertex prev;
    public int scratch;

    public Vertex(String nm){
        name=nm; adj=new LinkedList<>(); reset();
    }

    public void reset(){
        dist=Graph.INFINITY; prev=null; scratch=0;
    }
}

class Graph{
    public static final double INFINITY = Double.MAX_VALUE;

    public void addEdge(String sourceName, String destName, double cost){
        Vertex v = getVertex(sourceName);
        Vertex w = getVertex(destName);
        v.adj.add(new Edge(w, cost));
    }

    private Vertex getVertex(String vertexName) {
        Vertex v = vertexMap.get(vertexName);
        if (v == null){
            v = new Vertex(vertexName);
            vertexMap.put(vertexName, v);
        }
        return v;
    }
```

```
    private void printPath(Vertex dest){
        if (dest.prev != null) {
            printPath(dest.prev);
            System.out.print(" to ");
        }
        System.out.print(dest.name);
    }

    private void clearAll(){
        for (Vertex v : vertexMap.values())
            v.reset();
    }

    private Map<String, Vertex> vertexMap = new HashMap<String, Vertex>();

}
```

Vedlegg C: class GTree og class GNode

```
public class GTree<Anytype> {
    GNode root;

    class GNode {
        LinkedList<GNode> list;
        Anytype element;

        void add(Anytype e){
            add(new GNode(e));
        }

        void add(GNode n){
            list.add(n);
        }

    } // End of class GNode

    public GTree(Anytype e){ ... }
    public GTree(Anytype e, GtTree... gtrees){ ... }

    public void clear(){
        root = null;
    }
} // End of class GTree
```

Vedlegg D: Variabelt antall parametre

Metode som skal ta imot variabelt antall parametre har følgende syntaks:

```
return_type method_name(data_type... variableName){ }
```

Altså tre punktum for å vise at det kan komme null eller flere av denne datatypen. Inne i metoden kan parameterne aksesseres som en tabell. Eksempel:

```
static void display(String... values){
    for (String s:values){
        System.out.println(s);
}
```

Denne metoden kan eksempelvis kalles slik:

```
display();                  // Ingen parametre
display("God dag!");        // En parameter
display("God ", "dag!");    // To parametre
```