Høgskolen i Telemark

EKSAMEN

5610 ALGORITMAR OG DATASTRUKTURAR

20.05.2014

Tid:	9-14 (5 timar)
Målform:	Bokmål og nynorsk
Sidetal:	11 (forside + 5 + 5)
Hjelpemiddel:	Alle trykte og skrivne
Vedlegg:	Ingen

Eksamensresultata blir offentliggjort på nettet via Studentweb

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 - Algoritmeanalyse (20%)

Oppgave 1a (10%)

Hvilken orden har de følgende operasjonene? Dersom «Average Case» avviker fra «Worst Case» så oppgi begge. Gi *kort* begrunnelse for svarene.

1) Sette inn et element i en sortert pekerkjede (som LinkedList) med N elementer.
2) Sette inn et element i en sortert tabell (som ArrayList) med N elementer.
3) Innsetting i AVL-tre (med N elementer) kan beskrives i to faser: Først sette inn det nye elementet på sin rette plass, deretter kontrollere og gjenskape balanse. Hvilken orden har første fase – innsettingen?
4) Hvilken orden har andre fase av AVL-innsettingen – kontrollere og gjenskape balanse?
5) Innsetting av ny verdi i en Binary Heap med N elementer.

Oppgave 1b (10%)

For hver metode/kodebit: oppgi hvilken orden den har som funksjon av n, og gi *kort* forklaring/argumentasjon for dette.

1)

```
// Returnerer x opphøyd i n
public static int power(int x, int n){
  return (n == 0) ? 1 : x*power(x, n-1);
}
```

2)

```
// Returnerer x opphøyd i n
public static int power(int x, int n){
  if (n == 0) return 1;
  int y = power(x, n/2);
  return (n%2 == 0) ? y*y : y*y*x;
}
```

3)

```
// Utfører binærsøk iterativt på sortert tabell a[0..n-1]
int verdi = 1;
while(verdi < n)
```

```
{
  binærsøk(a, verdi);
  verdi *= 2;
}
```

4)

```
// Utfører søk iterativt på sortert tabell a[0..n-1]
int verdi = 0;
while(verdi < n)
{
  if (verdi%2 == 0)
    binærsøk(a, verdi);
  else
    lineærsøk(a, verdi);
  verdi++;
}
```

5)

```
int j = n;
while (j >= 1) {
  for (int i = 1; i <= j; i++ )
    sum++;
  j = j/2;
}
```

Oppgave 2 – Gap buffer (48%)

Hvilken datastruktur skal man bruke dersom man skal lage en enkel tekstbehandler som Notepad? Den må holde på tekst, som er en sekvens av tegn. Første tanke vil nok være å bruke pekerkjede (LinkedList) eller tabell (ArrayList). Men begge har sine ulemper.

Oppgave 2a (8%)

Forklar kort hvilke ulemper du ser ved å bruke en LinkedList eller ArrayList til å holde en sekvens av tegn i en tekstbehandler.

Oppgave 2b (8%)

Vi skal isteden bruke en datastruktur som går under navnet «Gap buffer». På skjermen ser brukeren teksten med en markør (cursor) plassert mellom to tegn, markøren viser hvor man kan slette eller sette inn tegn. Internt i programmet ligger teksten i en vanlig tabell av tegn, men teksten er delt i to – før og etter markøren. Tekst før markøren ligger fra starten av tabellen, tekst etter markøren ligger på slutten av tabellen. I mellom er det et «gap». Dersom markøren står helt først (bakerst) i teksten på skjermen, vil første (siste) del av teksten i tabellen være tom. Noen eksempler:

På skjermen er vist teksten «He|i, har du det bra?» og markøren står mellom «e» og «i». Internt i programmet ser tabellen slik ut:

H	e					i	,		h	a	r		d	u		d	e	t		b	r	a	?

Markøren er her representert som et «gap» mellom «e» og «i».

Bruker ønsker å endre «du» til «dere», og må flytte markøren til mellom «d» og «u». Da vil det se slik ut internt:

H	e	i	,		h	a	r		d					u		d	e	t		b	r	a	?

For hver posisjon markøren flyttes mot høyre, må altså første tegn i andre tekstdel flyttes og bli siste tegn i første tekstdel. Et tegn flyttes mot venstre når markøren/gapet flyttes mot høyre.

Så trykkes «Delete»-tasten. Da vil tabellen se slik ut:

H	e	i	,		h	a	r		d							d	e	t		b	r	a	?

Men! dersom du tar vare på første og siste indeks i gap-området, trenger du ikke fysisk slette tegnet, det er nok å øke siste-indeksen.

Så tastes tegnene «ere», dermed skal det se slik ut:

H	e	i	,		h	a	r		d	e	r	e				d	e	t		b	r	a	?

Hva skjer dersom bruker skriver videre, slik at gapet blir borte? Da må tabellen «utvides» slik som i ArrayList – opprett en ny, vesentlig større, og kopier over.

Klassen GapBuffer skal bare ha tre instansvariable: en tabell av char, en `int gapStart` som husker indeks til første posisjon i gapet, og en `int gapSlutt` som husker indeks til siste posisjon i gapet.
Videre skal den ha parameterløs konstruktørmetode, og konstruktørmetode som tar inn en String og setter markøren/gapet foran første tegn. Begge skal lage et gap på 10 tegn.
Lag klassen GapBuffer i samsvar med dette.

Oppgave 2c (8%)

Den interne hjelpemetoden sjekkBuffer() skal undersøke om gapet er tomt. I så fall skal det lages gap ved å lage nytt buffer som er 5% større enn det gamle, men samtidig skal gapet bli minst 10 tegn. **Lag metoden.**

Oppgave 2d (10%)

Klassen skal ha tre public metoder for å flytte markøren. Metoden høyre() skal flytte markøren én plass mot høyre. Metoden venstre() skal flytte markøren én plass mot venstre. Metoden flyttTil(int n) skal flytte markøren slik at den står foran tegn n, der tegnene er nummerert fra 0. **Lag disse tre metodene, vurder mulige feilsituasjoner.**

Oppgave 2e (8%)

Metoden delete() skal slette tegnet til høyre for markøren. Metoden backspace() skal slette tegnet til venstre for markøren. Metoden settInn(char c) skal sette inn nytt tegn ved markøren, og sette markøren etter det nye tegnet. **Lag disse tre metodene.**

Oppgave 2f (6%)

Klassen må reimplementere toString slik at den egentlige teksten som er representert, uten gapet, returneres. **Lag denne.**

Oppgave 3 – Lagerplanlegging (32%)

Du skal organisere et stort lager, der en mengde store, tunge kasser skal inn. Av sikkerhetsgrunner skal alle kassene stå på gulvet, de skal ikke stables på noe vis. Alle kassene er rektangulære, hver kasse har en viss lengde og bredde målt i desimeter (heltall). Høyden er ikke registrert siden de ikke skal stables, alle er lave nok til å gå inn i lagerhallen. Vi skal prøve å finne ut om alle kassene får plass. Kassene er representert ved objekter av class Kasse:

```
class Kasse {
  protected int id; //unik id for kassen
  protected int lengde, bredde; // Målt i desimeter
  // Konstruktørmetoder og set-/get-metoder er laget
}
```

Lagerhallen er representert ved et objekt av klassen Lagerhall:

```
class Lagerhall {
  protected int lengde, bredde; // Hallens totale lengde og
                                // bredde målt i desimeter
  public boolean plass(List<Kasse> kasser){...}
  // Evt. hjelpemetoder
}
```

Oppgave 3a (24%)

Skriv metoden

```
public boolean plass(List<Kasse> kasser)
```

i klassen Lagerhall som svarer på om det er plass til alle kassene. Hint: Opprett og vedlikehold en 2-dimensjonal tabell – et «kart» over hallen – og bruk rekursjon: backtracking. Lag de hjelpemetoder du måtte ønske. Aktuelle hjelpemetoder kan være å sjekke om bestemt kasse kan plasseres på bestemt sted, plassere kasse, og fjerne kasse.

Oppgave 3b (8%)

Du oppdager at det er ikke godt nok å vite **at** kassene går inn, du må vite **hvordan**! Forklar, med ord og/eller kodebiter, hvordan du vil endre programmet for å oppnå dette.

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 - Algoritmeanalyse (20%)

Oppgåve 1a (10%)

Kva for ein orden har dei fylgjande operasjonane? Dersom «Average Case» er ulik «Worst Case» så oppgje begge. Gje *kort* grunngjeving for svara.

1) Sette inn eit element i ei sortert peikarkjede (som LinkedList) med N element.
2) Sette inn eit element i ein sortert tabell (som ArrayList) med N element.
3) Innsetting i AVL-tre (med N element) kan beskrivast i to fasar: Fyrst sette inn det nye elementet på sin rette plass, deretter kontrollere og gjenskape balanse. Kva for ein orden har fyrste fase – innsettinga?
4) Kva for ein orden har andre fase av AVL-innsettinga – kontrollere og gjenskape balanse?
5) Innsetting av ny verdi i ein Binary Heap med N element.

Oppgave 1b (10%)

For kvar metode/kodebit: oppgje kva for orden den har som funksjon av n, og gje *kort* forklaring/argumentasjon for dette.

1)

```
// Returnerer x opphøgd i n
public static int power(int x, int n){
  return (n == 0) ? 1 : x*power(x, n-1);
}
```

2)

```
// Returnerer x opphøgd i n
public static int power(int x, int n){
  if (n == 0) return 1;
  int y = power(x, n/2);
  return (n%2 == 0) ? y*y : y*y*x;
}
```

3)

```
// Utfører binærsøk iterativt på sortert tabell a[0..n-1]
int verdi = 1;
while(verdi < n)
```

```
{
  binærsøk(a, verdi);
  verdi *= 2;
}
```

4)

```
// Utfører søk iterativt på sortert tabell a[0..n-1]
int verdi = 0;
while(verdi < n)
{
  if (verdi%2 == 0)
    binærsøk(a, verdi);
  else
    lineærsøk(a, verdi);
  verdi++;
}
```

5)

```
int j = n;
while (j >= 1) {
  for (int i = 1; i <= j; i++ )
    sum++;
  j = j/2;
}
```

Oppgåve 2 – Gap buffer (48%)

Kva for ein datastruktur skal ein nytte dersom ein skal lage ein enkel teksthandsamar som Notepad? Den må halde på tekst, som er ein sekvens av teikn. Fyrste tanke vil nok vera å bruke peikarkjede (LinkedList) eller tabell (ArrayList). Men begge har sine ulemper.

Oppgåve 2a (8%)

Forklar kort kva for ulemper du ser ved å bruke ein LinkedList eller ArrayList til å halde ein sekvens av teikn i ein teksthandsamar.

Oppgåve 2b (8%)

Vi skal istaden bruke ein datastruktur som går under namnet «Gap buffer». På skjermen ser brukaren teksten med ein markør (cursor) plassert mellom to teikn, markøren viser kvar ein kan slette eller sette inn teikn. Internt i programmet ligg teksten i ein vanleg tabell av teikn, men teksten er delt i to – før og etter markøren. Tekst før markøren ligg frå starten av tabellen, tekst etter markøren ligg på slutten av tabellen. I mellom er det eit «gap». Dersom markøren står helt fyrst (sist) i teksten på skjermen, vil fyrste (siste) del av teksten i tabellen vera tom. Nokre døme:

På skjermen er vist teksten «He|i, har du det bra?» og markøren står mellom «e» og «i». Internt i programmet ser tabellen slik ut:

H	e					i	,		h	a	r		d	u		d	e	t		b	r	a	?

Markøren er her representert som eit «gap» mellom «e» og «i».

Brukar ynskjer å endre «du» til «dere», og må flytte markøren til mellom «d» og «u». Då vil det sjå slik ut internt:

H	e	i	,		h	a	r		d					u		d	e	t		b	r	a	?

For kvar posisjon markøren flyttast mot høgre, må altså fyrste teikn i andre tekstdel flyttast og bli siste teikn i fyrste tekstdel. Eit teikn flyttast mot venstre når markøren/gapet flyttast mot høgre.

Så trykkast «Delete»-tasten. Då vil tabellen sjå slik ut:

H	e	i	,		h	a	r		d						d	e	t		b	r	a	?

Men! dersom du tek vare på fyrste og siste indeks i gap-området, treng du ikkje fysisk slette teiknet, det er nok å auke siste-indeksen.

Så tastast teikna «ere», dermed skal det sjå slik ut:

H	e	i	,		h	a	r		d	e	r	e			d	e	t		b	r	a	?

Kva skjer dersom brukar skriv vidare, slik at gapet blir borte? Då må tabellen «utvidast» slik som i ArrayList – opprett ein ny, vesentlig større, og kopier over.

Klassa GapBuffer skal berre ha tre instansvariable: ein tabell av char, ein `int gapStart` som hugsar indeks til fyrste posisjon i gapet, og ein `int gapSlutt` som hugsar indeks til siste posisjon i gapet.
Vidare skal den ha parameterlaus konstruktørmetode, og konstruktørmetode som tek inn ein String og sett markøren/gapet framom fyrste teikn. Begge skal lage et gap på 10 teikn.
Lag klassa GapBuffer i samsvar med dette.

Oppgåve 2c (8%)

Den interne hjelpemetoden sjekkBuffer() skal undersøke om gapet er tomt. I så fall skal det lagast gap ved å lage nytt buffer som er 5% større enn det gamle, men samtidig skal gapet bli minst 10 teikn. **Lag metoden.**

Oppgåve 2d (10%)

Klassa skal ha tre public metoder for å flytte markøren. Metoden høgre() skal flytte markøren ein plass mot høgre. Metoden venstre() skal flytte markøren ein plass mot venstre. Metoden flyttTil(int n) skal flytte markøren slik at den står framom teikn n, der teikna er nummerert frå 0. **Lag desse tre metodane, vurder mogelege feilsituasjonar.**

Oppgåve 2e (8%)

Metoden delete() skal slette teiknet til høgre for markøren. Metoden backspace() skal slette teiknet til venstre for markøren. Metoden settInn(char c) skal sette inn nytt teikn ved markøren, og sette markøren etter det nye teiknet. **Lag desse tre metodane.**

Oppgåve 2f (6%)

Klassa må reimplementere toString slik at den eigentlege teksten som er representert, utan gapet, returnerast. **Lag denne.**

Oppgåve 3 – Lagerplanlegging (32%)

Du skal organisere eit stort lager, der ei mengd store, tunge kasser skal inn. Av tryggleiksgrunnar skal alle kassene stå på golvet, dei skal ikkje stablast på noko vis. Alle kassene er rektangulære, kvar kasse har ei viss lengd og breidd målt i desimeter (heiltal). Høgda er ikkje registrert sidan dei ikkje skal stablast, alle er låge nok til å gå inn i lagerhallen. Vi skal prøve å finne ut om alle kassene får plass. Kassene er representert ved objekt av class Kasse:

```
class Kasse {
  protected int id; //unik id for kassa
  protected int lengd, breidd; // Målt i desimeter
  // Konstruktørmetoder og set-/get-metoder er laga
}
```

Lagerhallen er representert ved eit objekt av klassa Lagerhall:

```
class Lagerhall {
  protected int lengd, breidd; // Hallens totale lengd og
                               // breidd målt i desimeter
  public boolean plass(List<Kasse> kasser){...}
  // Evt. hjelpemetoder
}
```

Oppgåve 3a (24%)

Skriv metoden

```
public boolean plass(List<Kasse> kasser)
```

i klassa Lagerhall som svarer på om det er plass til alle kassene. Hint: Opprett og vedlikehald ein 2-dimensjonal tabell – eit «kart» over hallen – og bruk rekursjon: backtracking. Lag dei hjelpemetoder du måtte ynskje. Aktuelle hjelpemetoder kan vera å sjekke om bestemt kasse kan plasserast på bestemt stad, plassere kasse, og fjerne kasse.

Oppgåve 3b (8%)

Du oppdagar at det er ikkje godt nok å vite **at** kassene går inn, du må vite **korleis**! Forklar, med ord og/eller kodebitar, korleis du vil endre programmet for å oppnå dette.

Lykke Til!