NTNU
Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Eksamensoppgave i PROG1003
– Objekt-orientert programmering

Faglig kontakt under eksamen:	**Frode Haug**
Tlf:	**950 55 636**
Eksamensdato:	**7.juni 2021** **- HJEMME (A-F)**
Eksamenstid (fra-til):	**09:00-13:00 (4 timer)**
Hjelpemiddelkode/Tillatte hjelpemidler:	**F - Alle trykte og skrevne.** (kalkulator er *ikke* tillatt)
Annen informasjon:	
Målform/språk:	**Bokmål**
Antall sider (inkl. forside):	**6**

Informasjon om trykking av eksamensoppgaven

Originalen er:

1-sidig X **2-sidig** ☐

sort/hvit X **farger** ☐

Skal ha flervalgskjema ☐

Kontrollert av:

_____

Dato Sign

NB: Oppgave 1a, 1b, 1c, 1d, 2 og 3 er totalt uavhengige og kan derfor løses separat.

Oppgave 1 (40%)

Vi har følgende konvertering/sammenheng mellom siffer/tall og bokstav:

0 = A	1 = B	2 = C	3 = D	4 = E	5 = F	6 = G	7 = H	8 = I	9 = J
10 = K	11 = L	12 = M	13 = N	14 = O	15 = P	16 = Q	17 = R	18 = S	19 = T

a)

Vi har:

- `string tekst = "....."`;

  Der `"....."` totalt er på 30 tegn. Den er bygd opp av *to* ord, hvert på tre bokstaver, satt rett etter hverandre, og dette er gjentatt fem ganger.
  Tre-bokstavs ordene (norsk eller engelsk) skal du selv velge/finne.
  Det første ordet *skal* starte med bokstaven som tilsvarer det *nest siste* sifferet i kandidatnummeret ditt, og *bare* inneholde ulike bokstaver. Det andre ordet *skal* starte med bokstaven som tilsvarer det *siste* sifferet i kandidatnummeret ditt, og *bare* inneholde ulike bokstaver. Skulle disse to sifrene være like, får *ikke* ordene lov til å være like.
  F.eks: De to siste sifrene er «37». '3' tilsvarer 'D'. Velger ordet «DAG». '7' tilsvarer 'H'. Velger ordet «HÅR». Sammensatt blir dette «DAGHÅR».
  `tekst` blir da: «DAGHÅRDAGHÅRDAGHÅRDAGHÅRDAGHÅR»
  (*ikke* lov å velge disse ordene!)

- `int i = <nest siste siffer i kandidatnummeret>;`
- `int j = ( <siste siffer i kandidatnummeret> % 5 ) + 1;`

Skriv opp 6-bokstavsordet. Hva vil en løkke som starter på `i`, for hver gang øker med `j`, går *fem* ganger og inneholder koden: `cout << ' ' << tekst[i];` skrive ut?

b)

Vi har:

- en tom `string`
- en tom `<list>` med `int`
- `int sifre = <to siste sifre i kandidatnummeret>;`
- `int tall = sifre % 20;`
- `char tegn = char('A' + tall);`

1. Legg `tall` og dens fire etterfølgende heltallsverdier (totalt fem tall) inn i listen ved å utføre `push_back` på de tre første, og `push_front` på de to siste. F.eks. om `tall` er 44, så legges 44, 45, 46, 47 og 48 inn i listen.

2. Utfør `push_back` på stringen med `tegn` og deretter fortløpende dens fire etterfølgende bokstaver i alfabetet (totalt fem bokstaver).

3. **Skriv/oppgi:**
   - **stringens innhold**
   - **summen av elementene i listen**
   - **element/verdien til nr.2 og nr.4 i listen** (husk at den første er nr.1 og *ikke* nr.0)

c) (Dette er *ikke* et komplett kompilerende program):

```
1  map <string, By*> gByene;

2  int main()  {

3    By* nyBy = nullptr;

4    nyBy = new By("Hamar", 31369, 350.94);
5    gByene.insert(pair <string, By*> (nyBy->hentID(), nyBy));
6    nyBy = new By("Lillehammer", 28345, 478.16);
7    gByene.insert(pair <string, By*> (nyBy->hentID(), nyBy));

8    nyBy = new By("Gjovik", 30560, 672.25);
9    gByene[nyBy->hentID()] = nyBy;
10   nyBy = new By("Oslo", 693491, 454.12);
11   gByene[nyBy->hentID()] = nyBy;

12   for (const auto & val : gByene)
13       (val.second)->skrivData();

14   auto it = gByene.rbegin();
15   while (it != gByene.rend())
16         ((*it++).second)->skrivData();

17   auto it2 = gByene.find("Hamar");
18   if (it2 != gByene.end())  {
19      for_each(it2, gByene.end(), [] (const auto & val)
20              { cout << (val.second)->hentID()<< '\n'; } );
21   } else  cout << "\n\tIngen by heter 'Hamar'\n";

22   return 0;
   }
```

Kommenter med *egne ord* hva *hver eneste linje* gjør (henvis til linjenumrene 1-22).
Hva skrives *strukturmessig* ut, og forklar med *egne* ord hvorfor.
(`hentID()` returnerer byens navn (`string`))
(`skrivData()` skriver navn, antall innbyggere og flateinnhold (på ett eller annet format))

d) **Forklar med *egne ord* begrepene** (max. fire linjer pr. pkt):

1. Referanseoverføring
2. Overloading av funksjoner
3. Virtueller
4. Container. Nevn eksempler (både fra bibliotek og selvlagde) der dette er brukt i emnet.
5. Iterator

Oppgave 2 (40%)

Før du begynner å skrive kode: les *hele* teksten *nøye* for denne oppgaven.
Husk på, og bruk, funksjonene på `LesData2.h`**.**

Du skal lage *deler av* et program som holder orden på en persons lyspærer og stearinlys.

Vi forutsetter at følgende allerede er på plass/kodet:

- alle nødvendige includer
- `class Produsenter` - denne er *helt identisk* til `class Pasienter` i `EKS_28.CPP`, bare at den under `public` *kun* har funksjonene … `finn`(…) og …`hentNavn`(…). Klassen inneholder de *unike* navnene til *alle* produsenter av lyspærer og stearinlys (ikke noe i veien for at samme produsent lager begge deler)
- en «tradisjonell» `main` (i mange av eksemplene våre) som styrer programmet/kommandoene
- en `virtual void skrivData()` funksjon (inni hver av de tre klassene (se pkt.1-3 nedenfor) som skriver ut alle objektenes datamedlemmer (på en eller annen form).

Lag *komplett kode* for (eksakt navn på klasser og funksjoner bestemmer du selv):

1. **En baseklassen for begge typer lys**. Den *skal* inneholde:
   - to `private` datamedlemmer for produsentens nummer (refererer til den indeks i `Produsenter`) og antallet av lyset
   - en funksjon som endrer antall lys man har (øke med max.100 eller minske med max. så mange man har igjen)
   - en virtuell funksjon som leser inn og setter til alle datamedlemmene
   - en virtuell funksjon som skriver alle datamedlemmene til filen angitt av parameteren

2. **En avledet klasse (fra baseklassen) for lyspære**. Den *skal* inneholde:
   - to selvvalgte `private` datamedlemmer som er særegent for en lyspære (f.eks: type sokkel, om er en tradisjonell pære eller LED, matt eller klart glass, antall watt)
   - implementasjon av de to virtuelle funksjonene, og som begge også sørger for at tilsvarende funksjoner utføres i baseklassen

3. **En avledet klasse (fra baseklassen) for stearinlys**. Den *skal* inneholde det samme som den for lyspære, bare at datamedlemmene f.eks kan være dets lengde, tykkelse, ca.brennetid, farge, form

Du skal *ikke* lage constructorer/destructorer i noen av klassene.

4. ***En* global `vector`** som inneholder *pekere* til begge lystyper.
   ***Ett* globalt objekt** som inneholder alle (navnene på) produsentene.

5. **Fire funksjoner** (som kalles i/brukes av `main`) for å:

   - **skrive ut *alle* dataene om *alle* lysene**. Disse skal nummereres fra 1 og oppover. La det komme en egen melding om det er helt tomt for lys.

   - **legge inn *en* ny lystype** (lyspære eller stearinlys) bakerst i den globale vectoren. Brukeren spørres først om aktuell type ('P'(ære) eller 'S'(tearin)), så opprettes *ett* aktuelt nytt objekt, og *alle* dets data leses inn før innlegging. Til slutt skrives det nye lysets nummer ut.

   - **endre antallet som er igjen** av en gitt lystype (brukeren bestemmer selv hvilke *lovlig* lysnummer (i vectoren) som skal endres)

   - **skrive alle lysenes data til filen** «LYS.DTA». Velg selv formatet, men **oppgi det i besvarelsen**. Husk at det er to ulike objekttyper som skrives til fil, derfor må dette også angis på filen.

Oppgave 3 (20%)

1. **Definer en liste bestående av `string`**.
2. **Lag en funksjon, som mottar en liste som parameter**, og som via dette initierer den originale listen med ti selvvalgte *ulike* personnavn (*ett* ord). Navnene skal *ikke* komme i noen sortert/bestemt rekkefølge. De fem første legges inn bakerst, det fem siste legges inn forrest.

Det skal nå videre opereres på listen laget i pkt. nr.1, og initiert ved kall på funksjonen i pkt. nr. 2.

Dere skal *kun* gjøre *en* av de fire blokkene (A, B, C, D) nedenfor, ut i fra sifre i kandidatnummeret deres:

A) ***Siste siffer er partall (0, 2, 4, 6, 8) og nest siste siffer er også partall:***

3. Skriv ut hele listens innhold ved å bruke range-based for-løkke
4. Tell opp antallet (og skriv ut) vha. `for_each(…)` fra `<algorithm>` (*ikke* bruk `size()`)
5. Skriv ut verdien til det *første* elementet ved å bruke en funksjon i `<list>`
6. Skriv ut verdien til det *siste* elementet ved å bruke en iterator
7. Bruk funksjon fra `<algorithm>` for å finne ut om den *ikke er* sortert. Skrive i så fall ut en tekst om at det *er* den *ikke.*
8. Bruk funksjon fra `<algorithm>` til å finne (og skrive ut) det *minste* elementet
9. Slett det *andre* elementet *bakfra*
10. Sett inn et nytt element (navn) som nr.3 *forfra*
11. Bruk funksjon fra `<algorithm>` for å sette en iterator (og evt. skrive ut) til det første navnet *startende* på 'A'
12. Reverser (snu baklengs) lista ved å bruke en funksjon i `<list>`
13. Sorter lista
14. Bruk funksjon fra `<algorithm>` for å sette en iterator til et element som er det første som er *større enn* et gitt navn
15. Bruk funksjon fra `<algorithm>` for å sette en iterator til et element som er *større eller lik* et gitt navn. Skriv en egen melding om en slik *ikke* ble funnet.

B) ***Siste siffer er partall (0, 2, 4, 6, 8) og nest siste siffer er oddetall (1, 3, 5, 7, 9):***

3. Skriv ut hele listens innhold ved å bruke `for_each(…)` fra `<algorithm>`
4. Tell opp antallet (og skriv ut) vha. range-based for-løkke (*ikke* bruk `size()`)
5. Skriv ut verdien til det *første* elementet ved å bruke en iterator
6. Skriv ut verdien til det *siste* elementet ved å bruke en funksjon i `<list>`
7. Bruk funksjon fra `<algorithm>` for å finne ut om den *er* sortert. Skrive i så fall ut en tekst om at det *er* den.
8. Bruk funksjon fra `<algorithm>` til å finne (og skrive ut) det *største* elementet
9. Slett det *andre* elementet *forfra*
10. Sett inn et nytt element (navn) som nr.3 *bakfra*
11. Bruk funksjon fra `<algorithm>` for å få tak i (og skrive ut) *antall* navn *startende* på 'A'
12. Reverser (snu baklengs) lista ved å bruke en funksjon i `<algorithm>`
13. Sorter lista
14. Bruk funksjon fra `<algorithm>` for å sette en iterator til et element som er *større eller lik* et gitt navn
15. Bruk funksjon fra `<algorithm>` for å sette en iterator til et element som er det første som er *større enn* et gitt navn. Skriv en egen melding om en slik *ikke* ble funnet.

C) ***Siste siffer er oddetall (1, 3, 5, 7, 9) og nest siste siffer er også oddetall:***

3. Skriv ut hele listens innhold ved å bruke range-based for-løkke
4. Tell opp antallet (og skriv ut) vha. `for_each(…)` fra `<algorithm>` (*ikke* bruk `size()`)
5. Skriv ut verdien til det *første* elementet ved å bruke en iterator
6. Skriv ut verdien til det *siste* elementet ved å bruke en funksjon i `<list>`
7. Bruk funksjon fra `<algorithm>` for å finne ut om den *ikke er* sortert.
   Skrive i så fall ut en tekst om at det *er* den *ikke.*
8. Bruk funksjon fra `<algorithm>` til å finne (og skrive ut) det *minste* elementet
9. Slett det *andre* elementet *forfra*
10. Sett inn et nytt element (navn) som nr.3 *bakfra*
11. Bruk funksjon fra `<algorithm>` for å sette en iterator (og evt. skrive ut)
    til det første navnet *startende* på 'A'
12. Reverser (snu baklengs) lista ved å bruke en funksjon i `<list>`
13. Sorter lista
14. Bruk funksjon fra `<algorithm>` for å sette en iterator til et element
    som er *større eller lik* et gitt navn
15. Bruk funksjon fra `<algorithm>` for å sette en iterator til et element som er
    det første som er *større enn* et gitt navn. Skriv en egen melding om en slik *ikke* ble funnet.

D) ***Siste siffer er oddetall (1, 3, 5, 7, 9) og nest siste siffer er partall (0, 2, 4, 6, 8):***

3. Skriv ut hele listens innhold ved å bruke `for_each(…)` fra `<algorithm>`
4. Tell opp antallet (og skriv ut) vha. range-based for-løkke (*ikke* bruk `size()`)
5. Skriv ut verdien til det *første* elementet ved å bruke en funksjon i `<list>`
6. Skriv ut verdien til det *siste* elementet ved å bruke en iterator
7. Bruk funksjon fra `<algorithm>` for å finne ut om den *er* sortert.
   Skrive i så fall ut en tekst om at det *er* den.
8. Bruk funksjon fra `<algorithm>` til å finne (og skrive ut) det *største* elementet
9. Slett det *andre* elementet *bakfra*
10. Sett inn et nytt element (navn) som nr.3 *forfra*
11. Bruk funksjon fra `<algorithm>` for å få tak i (og skrive ut) *antall* navn *startende* på 'A'
12. Reverser (snu baklengs) lista ved å bruke en funksjon i `<algorithm>`
13. Sorter lista
14. Bruk funksjon fra `<algorithm>` for å sette en iterator til et element
    som er det første som er *større enn* et gitt navn
15. Bruk funksjon fra `<algorithm>` for å sette en iterator til et element
    som er *større eller lik* et gitt navn. Skriv en egen melding om en slik *ikke* ble funnet.

Annet (klargjørende):

- *Ikke* bruk saker i STL utenfor pensumet, templates eller annet stoff/biblioteker utenfor pensum.
- Gjør dine egne forutsetninger og presiseringer av oppgaven, dersom du skulle finne dette nødvendig. Gjør i så fall klart rede for disse *i starten* av din besvarelse av oppgaven.

Lykke til med *eget, selvstendig og individuelt* arbeid!

FrodeH