Institutt for datateknologi og informatikk

Eksamen TDT4102 - Prosedyre- og objektorientert programmering

Eksamensdato : Fredag 7. august 2020.
Eksamenstid (fra-til) : 0900-1300
Hjelpemiddelkode/Tillatte hjelpemiddel: A / Alle hjelpemiddel tillat
Faglig kontakt under eksamen : Rune Sætre and Truls Asheim.
Backup (Özlem Özgöbek)
Tlf : 452 18 103 (Rune), +45 2282 5830 (Truls), 457 90 081 (Özlem)
Email: tdt4102-fagans@idi.ntnu.no
Teknisk hjelp under eksamen: NTNU Orakel. Tlf : 73 59 16 00

ANNEN INFORMASJON
Gjør dine egne antagelser og presiser i besvarelsen hvilke forutsetninger du har lagt til grunn i tolking/avgrensing av oppgaven. Faglig kontaktperson skal kun kontaktes dersom det er direkte feil eller mangler i oppgavesettet.

Lagring:
Besvarelsen din i Inspera Assessment lagres automatisk hvert 15. sekund. Mens du jobber i et annet program – husk å lagre underveis.

Juks/plagiat:
Eksamen skal være et individuelt, selvstendig arbeid. Det er tillatt å bruke hjelpemidler. Alle besvarelser blir kontrollert for plagiat. Du kan lese mer om juks og plagiering på eksamen her: https://innsida.ntnu.no/wiki/-/wiki/Norsk/Juks+påeksamen

Kildehenvisninger:
Selv om "Alle hjelpemiddel er tillatt", er det ikke tillatt å kopiere andres kode og levere den som din egen. Du kan se på andre åpent tilgjengelige ressurser, og deretter skrive din egen versjon av det du så, i henhold til copyright-forskrifter.

Varslinger:
Hvis det oppstår behov for å gi beskjeder til kandidatene underveis i eksamen (for eksempel ved feil i oppgavesettet), vil dette bli gjort via varslinger i Inspera. Et varsel vil dukke opp som en dialogboks på skjermen i Inspera. Du kan finne igjen varselet ved å klikke på bjella øverst i høyre hjørne på skjermen. Det vil i tillegg bli sendt SMS til alle kandidater for å sikre at ingen går glipp av viktig informasjon. Ha mobiltelefonen din tilgjengelig.

Vekting av oppgavene:
Del 1 og Del 2 teller omtrent 25% hver av totalen, Del 3 teller da omtrent 50% av eksamen. Du vil allikevel ikke kunne få bestått på eksamen bare ved å svare perfekt på de to første delene.

Filopplasting:
Alle filer må være lastet opp i besvarelsen før eksamenstida går ut. Det er lagt til 15 minutter eksamenstid for digitalisering av håndtegninger/filer. Tilleggstida inngår i gjenstående eksamenstid som vises øverst til venstre på skjermen. Pass på at det ikke finnes noe forfatterinformasjon i filen(e) du skal levere.

I siste del av eksamen skal du bruke samme program som du satte opp i øving 0. Du må også vite hvordan du laster ned, pakker ut, og setter opp mapper fra en zip-fil som VS-code (eller tilsvarende) C++ prosjekter på maskinen din. Til slutt må du være i stand til pakke alle slike mapper sammen i en zip-fil igjen for å levere det du har kodet, innen tidsfristen, for å kunne bestå denne eksamen.

OM LEVERING:
Besvarelsen din leveres automatisk når eksamenstida er ute og prøven stenger, forutsatt at minst én oppgave er besvart. Dette skjer selv om du ikke har klikket «Lever og gå tilbake til Dashboard» på siste side i oppgavesettet. Du kan gjenåpne og redigere besvarelsen din så lenge prøven er åpen. Dersom ingen oppgaver er besvart ved prøveslutt, blir ikke besvarelsen din levert. Vær sikker på at du laster opp korrekt oppdatert zip-fil (Del 3) minst en gang ekstra, halvveis i eksamen.

Trekk fra eksamen:

Ønsker du å levere blankt/trekke deg, gå til hamburgermenyen i øvre høyre hjørne og velg «Lever blankt».
Dette kan ikke angres selv om prøven fremdeles er åpen.

Tilgang til besvarelse:

Du finner besvarelsen din i Arkiv etter at sluttida for eksamen er passert.

Vær så snill å les igjennom informasjonen til venstre nøye.

Hint: Du kan gjøre vinduet til venstre mye større, og fjerne små-side-visningen (sidestolpen) helt til venstre (øverst).

REGLER OG SAMTYKKER

Dette er en **individuell** øving. Du har ikke lov til å kommunisere (gjennom web-forum, chat, telefon, hverken i skriftlig, muntlig eller annen form), ei heller samarbeide med noen andre under eksamen.

Før du kan fortsette til selve øvingen må du forstå og SAMTYKKE i følgende:

Under øvingen:

Jeg skal IKKE motta hjelp fra andre.

- Aksepter

Jeg skal IKKE hjelpe andre eller dele løsningen min med noen.

- Aksepter

Jeg skal IKKE copy-paste noe kode fra noen eksisterende online/offline kilder. (Du kan se, og deretter skrive din EGEN versjon av koden).

- Aksepter

Jeg er klar over at øvingen kan bli underkjent uavhengig av hvor korrekt svarene mine er, hvis jeg ikke følger reglene og/eller IKKE aksepterer disse utsagnene.

- Aksepter

1(a) Se på følgende kode

```
1  map<string, string> people {{"Ola", "Nordmann"},
2                              {"Kari, Nordmann"},
3                              {"Lisa", "Andersen"}};
4  people.insert({"Ola", "Andersen"});
5  for (auto i : people){
6      cout << i.first << " " << i.second << endl;
7  }
```

Hva blir skrevet ut? Velg riktig rekkefølge.

(Ola Nordmann, Ola Andersen, Kari Nordmann, Lisa Andersen)

(Ola Andersen, Ola Nordmann, Kari Nordmann, Lisa Andersen)

(Ola Nordmann, Kari Nordmann, Ola Andersen, Lisa Andersen)

Hvis linje 4 så blir erstattet med

```
people["Ola"] = "Andersen";
```

hva blir skrevet ut? Velg riktig rekkefølge.

(Kari Nordmann, Ola Andersen, Ola Nordmann, Lisa Andersen)

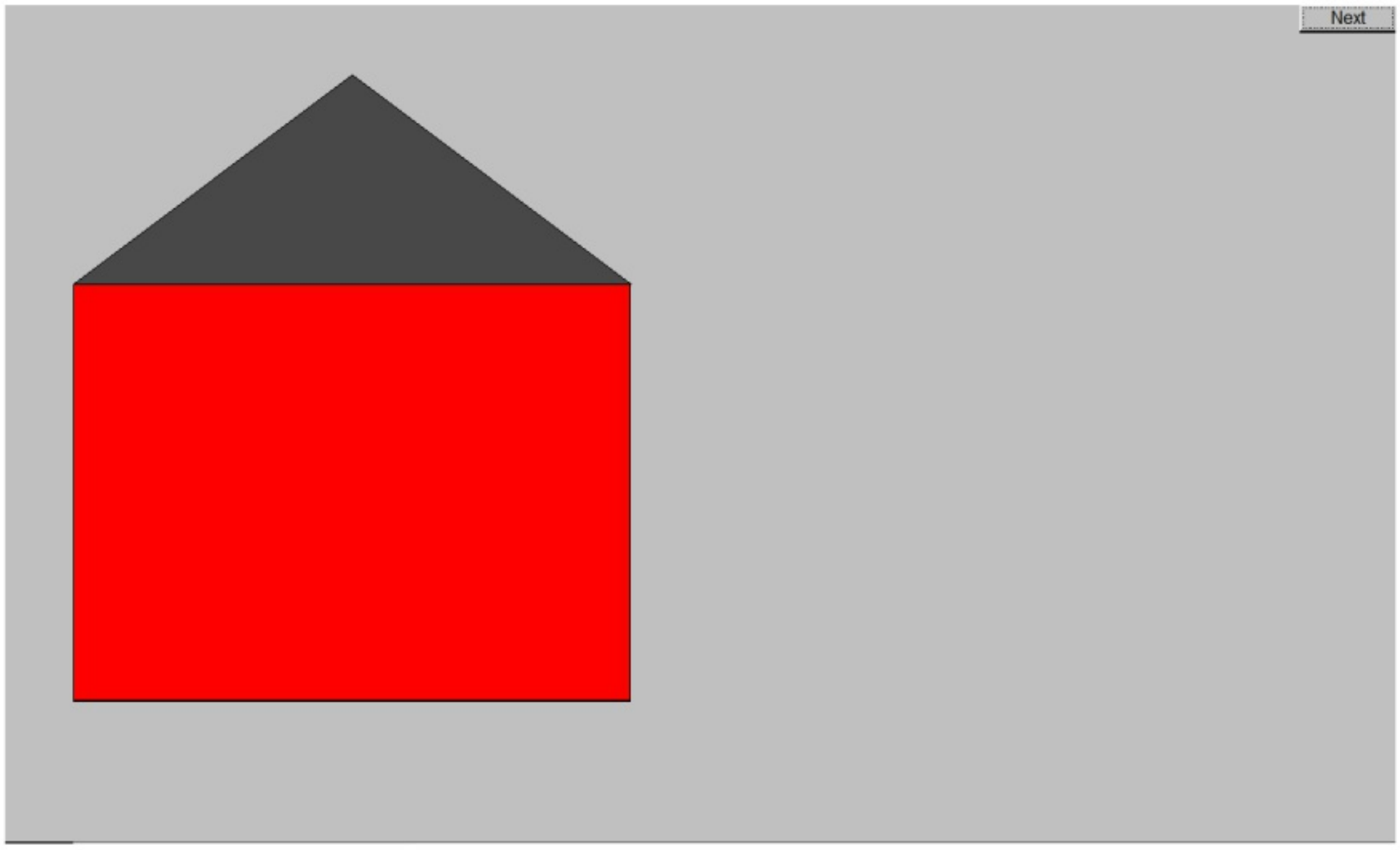
(Lisa Andersen, Ola Andersen, Ola Nordmann, Kari Nordmann)

(Kari Nordmann, Ola Andersen, Ola Nordmann, Lisa Andersen)

Maks poeng: 10

1(b) Vi ønsker lage følgende tegning:

Dessverre ser det ut til at noen deler av tegningen er borte:



Se på den følgende koden:

```
1  int main(){
2          using namespace Graph_lib;
3          Simple_window win{{0,0}, 1000, 600, "Window"};
4          Rectangle door {{250,300}, 100, 200};
5          door.set_fill_color(Color::black);
6          Rectangle window{{100, 250}, 75, 75};
7          window.set_fill_color(Color::blue);
8          Rectangle house{{50, 200}, 400, 300};
9          house.set_fill_color(Color::red);
10         Polygon roof;
11         roof.add({50, 200});
12         roof.add({450, 200});
13         roof.add({250, 50});
14         roof.set_fill_color(Color::dark_gray);
15         win.attach(door);
16         win.attach(window);
17         win.attach(house);
18         win.attach(roof);
19         win.wait_for_button();
20 }
```

Ved å flytte bare en linje til et annet sted i programmet kan du få tegningen til å bli riktig.
Hvilken linje vil du flytte?

Skriv ett linjenummer (mellom 1 og 20):

Hvor kan linjen flyttes til for å lage ønsket tegning?

Velg ett eller flere alternativer

- Mellom linje 1 og 2
- Mellom linje 2 og 3
- Mellom linje 3 og 4
- Mellom linje 9 og 10
- Mellom linje 14 og 15
- Mellom linje 19 og 20

Maks poeng: 10

1(c) Hva blir skrevet ut når følgende kode kjøres?

```
1  void bad_idea(vector<int> &vec) {
2    for (vector<int>::iterator it = vec.begin(); it != vec.end(); it++) {
3      if (vec.size() > 5) {
4        vec.pop_back();
5      }
6      cout << *it;
7    }
8  }
9
10 int main() {
11   vector<int> vec1 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
12   vector<int> vec2 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11};
13
14   bad_idea(vec1);
15   cout << endl;
16 }
```

Hva vil skje dersom vi erstatter vec1 med vec2 i kallet til funksjonen bad_idea på linje 14? Forklar kort hvorfor.

Maks poeng: 10

1(d) Marker følgende påstander som sant/usant:

Et map kan ha like nøkler, men ikke like verdier.

- Sant
- Usant

Aksessering av verdier i et map ved bruk av medlemsfunksjonen at() skiller seg fra [] ved at et unntak kastes når man bruker en nøkkel som ikke fins i mapet som argument.

- Usant
- Sant

Et map sorterer automatisk elementene i stigende rekkefølge.

- Usant
- Sant

Fordelen med å bruke insert() fremfor [] når man skal legge til elementer i et map er at insert() kan duplisere eksisterende nøkler hvis man legger til et element med en nøkkel som allerede fins.

- Usant
- Sant

Det regnes som god skikk å definere maps i headerfiler slik at man enkelt kan få tilgang til det.

- Sant
- Usant

Maks poeng: 10

1(e) Du ønsker å dividere heltall og lager en enkel funksjon for dette:

```
1  double divideFunc(int a, int b){
2      return static_cast<double>(a/b);
3  }
```

Du er klar over at ikke alle heltallsdivisjoner gir heltall, du har derfor valgt å konvertere divisjonen til en double vha. static_cast<double>, men når du tester funksjonen ender du alltid opp med heltall likevel. Hvorfor skjer det?

Hvordan kan du IKKE fikse denne feilen? Velg feil alternativet.

- double **divideFunc**(int a, int b){
  return double static_cast(a)/static_cast(b); };
- double **divideFunc**(int a, int b){
  return (static_cast<double>(a)/b); };
- double **divideFunc**(int a, int b){
  return ((a)/static_cast<double>(b)); };
- double **divideFunc**(int a, int b){
  return (static_cast<double>(a)/static_cast<double>(b)); };

Maks poeng: 10

1(f)

```
1  template<class T>
2  void eraseFromList(list<T>& l, T elem){
3      for (auto it = l.begin(); it != l.end(); it++){
4          if (it == elem){
5              l.erase(elem);
6              return;
7          }
8      }
9  }
```

Det er to feil i koden over. På hvilke to linjer oppstår feilene?

Velg ett alternativ

- Linje 3 & 5
- Linje 3 & 4
- Linje 4 & 5
- Linje 2 & 5

Erstatt de to linjene med korrekt kode.

Linje nummer: Korrekt kode:

Linje nummer: Korrekt kode:

Maks poeng: 10

1(g) Hva er sant om callback-funksjonen til en knapp?
Velg ett alternativ:

- En callback-funksjon blir kalt av GUI når systemet oppdager et klikk på knappen
- En callback-funksjon kan ikke kalle andre funksjoner.
- Callback-funksjonen blir ikke kalt dersom et annet objekt er plassert over den tilhørende knappen i vinduet
- En callback-funksjon tar inn to adresser som argumenter og returnerer en peker.

Maks poeng: 10

1(h) Før dette årets Norgesferie ville du lage en lenket liste for å holde kontroll på turen og alle destinasjonene. Du lagde structen Destination, vist under, og initialiserte de første tre destinasjonene, Bergen, Voss og Flåm (i den rekkefølgen).

```
struct Destination{
string name;
Destination* next;
Destination* prev;
Destination(string name, Destination* next = nullptr, Destination* prev = nullptr) :
    name{name},
    next{next},
    prev{prev}{}
};

Destination* trip = new Destination{"Bergen"};
trip->next = new Destination{"Voss", nullptr, trip};
trip = trip->next;
trip->next = new Destination{"Flåm", nullptr, trip};
trip = trip->prev;
```

Hvilken av de følgende kodene vil korrekt skrive ut alle de tre destinasjonene i rett rekkefølge på turen?

Velg ett eller flere alternativer

```
Destination *current = trip;
do{
    cout << current -> name << '\n';
    current = current -> next;
} while (current);
```

```
Destination *current = trip;
while ((*current).next != nullptr){
    cout << (*current).name << '\n';
    current = (*current).next;
};
```

```
Destination* current = trip;
while (true){
    cout << current -> name << '\n';
    current = current -> next;
    if (current -> next) break;
}
```

```
Destination* current = trip -> prev;
while (current){
    current = current -> next;
    cout << current -> prev -> name << '\n';
};
```

Maks poeng: 10

1(i) Hvilken av de følgende linjene (nummer) viser en ugyldig metode for å legge til et element til vr.

```
1  Vector_ref<Shape> vr;
2  Circle cir1{Point{100, 100}, 40};
3  Circle* cir2 = new Circle{Point{300,300}, 50};
4
5  vr.push_back(cir1);
6
7  vr.push_back(&cir1);
8
9  vr.push_back(new Circle{Point{200,100}, 40});
10
11 vr.push_back(new Rectangle{Point{200,200}, Point{400,400}});
12
13 vr.push_back(&cir2);
14
15 vr.push_back(cir2);
16
17 vr.push_back(*cir2);
```

Skriv linje nummeret mellom 1 og 17:

Forklar hvorfor:

Maks poeng: 10

1(j) En restaurant har laget en funksjon for å reservere bord, som returnerer antall ledige bord etter reservasjonen. For å få til dette lagde de en global variabel, numberOfTables.

```
1  int numberOfTables = 20;
2
3  int reserveTables(int n){
4      if (n > numberOfTables){
5          cout << "Not enough available tables";
6          assert(numberOfTables >= 0);
7          return numberOfTables;
8      }
9      numberOfTables -= n;
10     assert(numberOfTables >= 0);
11     return numberOfTables;
12 };
```

Hva gjør funksjonen assert i linje 5 og 10?

Velg ett alternativ

- assert terminerer programmet dersom uttrykket er false.
- assert skriver ut en melding til brukeren hvis uttrykket er false, og fortsetter deretter programmet.
- assert vil omgjøre den siste operasjonen hvis uttrykket er false.
- assert sjekker om uttrykket er false og setter numberOfTables til en gyldig verdi.

Finn en løsning slik at de ikke trenger å bruke en global variabel for det totale antallet bord og forklar (max 200 tegn).

Maks poeng: 10

2(a)

```
1  namespace First{
2      int a;
3  };
4  namespace Second{
5      int a;
6  };
```

Basert på disse to navnerommene skal du implementere en main()-funksjon. Den skal inneholde følgende operasjoner:

- Lag en peker til variabelen a i navnerommet First. Denne pekeren kan hete ptr.
- Sett variabelen a i navnerommet First til å bli 5 ved å bruke pekeren ptr.
- Få pekeren ptr til å peke på a i navnerommet Second.
- Sett variabelen a i navnerommet Second til å bli 6 ved å bruke pekeren ptr.

Skriv ditt svar her

```
1
```

Maks poeng: 10

2(b) Se på følgende klasse:

```
1  class Student{
2          string name;
3          int studNr;
4  };
```

Du ønsker å overlaste operatoren «, slik at den skriver ut følgende: ”Name: <name>. Student number: <studNr>”. Er det mulig hvis operatoroverlastingen ikke settes som friend av klassen? Hvorfor/hvorfor ikke?

Implementer operatoroverlastingen beskrevet ovenfor.

Skriv ditt svar her

```
1
```

Maks poeng: 10

2(c) Den følgende funksjonen kalkulerer ønsket opphøying av tall. Skriv om funksjonens deklarasjon så den returnerer tallet 2 dersom den kalles uten noen argumenter.

```
int powerFunc(char c, int i, int j){
    int res = 0;
    switch(c){
    case 'a':
            res = pow(i, 2);
            break;
    case 'b':
            res = pow(i,3);
            break;
    case 'c':
            res = pow(i,j);
            break;
    default:
    }
    return res;
}
```

Skriv ditt svar her

2(d)

```
1  string s = "Hei";
2  string* sPtr = &s;
3  sPtr->push_back("Hallo");
```

Hvorfor blir koden over feil? Forklar.

Endre koden slik at sPtr sørger for at s får verdien ”HeiHallo”.

Skriv ditt svar her

```
1
```

Maks poeng: 10

2(e)

```
1  string get_feedback(unsigned int record) {
2    unsigned int score = 0;
3    for (;;) {
4      cout << "Enter a score (0-1000)";
5      cin >> score;
6      if (score <= 1000 && score >= 0) {
7        string feedback;
8        if (score > record) {
9          feedback = "Congratulations! You beat the record.";
10       } else {
11         feedback = "You didn't beat the record.";
12       }
13       break;
14     }
15   }
16   return feedback;
17 }
18
19 int main() {
20   cout << get_feedback(850) << endl;
21   return 0;
22 }
```

Koden over kan ikke kompileres. Kompilatoren sier at variabelen feedback på linje 16 er udefinert. Hvorfor skjer dette? (max 200 tegn)

Maks poeng: 10

2(f)

Du bruker en stor vektor med heltall i en funksjon, og ønsker ikke å lage kopier av den. Men du er redd for å utilsiktet endre uønsket på vektoren i funksjonen din.

```
1  void someFunc(vector<int> myVector);
```

Endre funksjonsdeklarasonen for å unngå kopiering av den store vektoren og samtidig sørge for at den ikke endres i funksjonen.

Skriv ditt svar her

```
1
```

Maks poeng: 10

2(g)

```
1  void insertLineNumbers(string filename){
2      ofstream outputFile{filename};
3      if (!outputFile){
4          error("Can not open file");
5      }
6      map<int, string> intToString{
7          {1, "First"},
8          {2, "Second"},
9          {3, "Third"},
10         {4, "Fourth"},
11         {5, "Fifth"},
12         {6, "Sixth"},
13         {7, "Seventh"},
14         {8, "Eighth"},
15         {9, "Ninth"},
16         {10, "Tenth"}
17     };
18     // Fill in code here:
19
20 }
```

Funksjonen over er nesten ferdig, men mangler hovedfunksjonaliteten sin. Den skal ta inn en tom tekstfil og fylle inn ti linjer: På første linje skal det stå ”First line”, på andre linje skal det stå ”Second line” og så videre helt til ti linjer. Fyll inn kode slik at dette blir løst ved å bruke mapet og en for-løkke.

Skriv ditt svar her

```
1
```

Maks poeng: 10

2(h)

```
1  void dynamicFunc(int n){
2      int* p1 = new int{n};
3      int* p2 = new int[n];
4      for (int i = 0; i < n; i++){
5          p2[i] = n;
6      }
7  };
```

Gitt funksjonen dynamicFunc(int n), skriv ferdig funksjonen slik at det ikke er noe minnelekkasje i funksjonen.

Skriv ditt svar her

```
1
```

Maks poeng: 10

2(i)

```
1  class C{
2  private:
3      vector<int*> values;
4      int val;
5  public:
6      C(int n){
7          for (int i = 0; i < n; i++){
8              int* value = new int{0};
9              values.push_back(value);
10         }
11     }
12     C(int n, int val) : C(n) {
13         for (auto el : values){
14             *el = val;
15         }
16     }
17     ~C();
18 };
```

Implementer destruktøren til klassen C og **forklar** hvilken type konstruktør C(int n, int val) er.

Skriv ditt svar her

```
1
```

Maks poeng: 10

2(j)

```
1  class Base{
2      private:
3          int num;
4      public:
5          Base(int n);
6  };
7
8  class Derived : public Base{
9      private:
10         string* sPtr;
11     public:
12 };
```

Koden over viser en klasse Derived som arver fra klassen Base. Derived skal ha en konstruktør som tar inn et heltall n og en peker til en streng, ptr. Konstruktøren til Derived skal sørge for at konstruktøren til Base blir kjørt med n som argument, før sPtr blir satt lik ptr. Hvis konstruktøren kun kalles med første argument, skal sPtr bli en nullptr. **Implementér** konstruktøren ved å kun bruke initialiseringsliste.

Skriv ditt svar her

```
1
```

Maks poeng: 10

VIKTIG:

Zip-filen du skal laste ned inneholder kompilerbare (og kjørbare) .cpp- og .h-filer med forhåndskodede deler, en datamappe med datafiler, og en full beskrivelse av alle oppgavene som en PDF-fil. Etter å ha lastet ned zip-filen står du fritt til å bruke et utviklingsmiljø etter eget valg (for eksempel VScode etc.) for å jobbe med oppgavene.

For å få bestått på denne eksamen er det **HELT AVGJØRENDE AT DU LASTER OPP ZIP-FILEN**, innen de 4 timene til rådighet. Ikke vent til siste minutt med å laste opp zip-filen!

De korte svarene i Inspera (del 1 og 2) lagres automatisk hvert 15. sekund, helt til eksamenstiden er slutt. Og zip-filen (i del 3) kan også endres / lastes opp flere ganger. Så hvis du vil gjøre noen endringer etter at du har lastet opp filen, kan du bare endre koden, zippe den sammen på nytt, og laste opp filen igjen. Når prøvetiden er over, vil siste versjon av alt du har skrevet eller lastet opp i Inspera automatisk bli sendt inn som ditt gjeldende svar, så **sørg for at du laster opp minst en gang halvveis, og en gang før tiden går ut.**

På neste side kan du laste ned .zip-filen, og senere laste opp den nye .zip-filen med din egen kode inkludert. Husk at PDF-dokumentet med alle oppgavene er inkludert i zip-filen du laster ned.

3(a) **LAST NED**

Trykk her for filnedlasting

LAST OPP

Last opp all den komplette koden som en .zip-fil. Ikke endre den opprinnelige mappestrukturen. For å få bestå prøven er det **HELT AVGJØRENDE AT DU LASTER OPP DEN NYE ZIP-FILEN DIN KORREKT**, minst en gang i løpet av de 4 timene til rådighet.

Det er mulig å oppdatere både enkelt-svarene og filopplastningen **flere ganger**, i tilfelle du retter på noe etter første innlevering.

Prøv å last opp en oppdatert zip-fil minst en gang ekstra, midt i prøvetiden, for å se at du klarer det, og for å finne ut hvor lang tid du bruker på det.

Last opp zip-filen med besvarelsen din her. Alt i én zip-fil.

Last opp filen her. Maks én fil.

Alle filtyper er tillatt. Maksimal filstørrelse er **50 GB**.

Velg fil for opplasting

Maks poeng: 120

Document 2
Attached

TDT4102 - Procedural and Object-Oriented Programming

7 August 2020

0.1 Generell Intro

Les gjennom oppgavetekstene nøye. Noen av oppgavene har lengre tekst, men dette er for å gi kontekst, introduksjon og eksempler til oppgavene.

Når det står "implementer", "definer" eller "lag" skal du skrive en fungerende implementasjon: hvis det handler om en funksjon skal du skrive deklarasjonen med returtype og parametertype(r) og hele funksjons-kroppen.

Når det står "deklarer" er vi kun interessert i funksjons- eller klassedeklarasjonen. Typisk vil dette være deklarasjoner du vanligvis finner i header-filer.

Hvis det står "forklar" står du fritt i hvordan du svarer, men bruk enkle kodelinjer og/eller korte tekstforklaringer og vær kort og presis.

Dersom du mener at opplysninger mangler i en oppgaveformulering, gjør kort rede for de antagelser og forutsetninger du finner nødvendig.

Hvis du oppdager at informasjon er feil eller mangler i en oppgave, forklar hvilke antagelser du må gjøre, og hvordan du tolker oppgaven. Legg forklaringene til som kommentarlinjer i koden din.

Hver enkelt oppgave er ikke ment å være mer omfattende enn det som er beskrevet. Noen oppgaver fokuserer bare på enkeltfunksjoner og da er det utelukkende denne funksjonen som er tema. Andre oppgaver er "oppskriftsbasert" og vi spør etter funksjoner som utgjør deler i et program, eller forskjellige deler av en eller flere klasser. Du kan velge selv om du vil løse dette trinnvis, eller om du vil lage en samlet implementasjon, men sørg for at det går tydelig frem hvilke spørsmål du har svart på hvor i koden din.

Husk at funksjonene du lager i en deloppgave ofte er ment å skulle brukes i andre deloppgaver. Selv om du står helt fast på en deloppgave bør du likevel prøve å løse alle eller noen av de etterfølgende oppgavene ved å anta at funksjoner fra tidligere deloppgave er riktig implementert.

All kode skal være i C++. Du kan bruke VS-code, eller andre utviklingsmiljøer om du ønsker. Oppgaven krever ikke kjennskap til andre klasser og funksjoner enn de du har blitt godt kjent med i øvingsopplegget, eller som står beskrevet i læreboka. Hvis du mener det er nødvendig kan du fritt inkludere flere klasser og funksjoner.

"include"-setninger og organisering av koden i flere filer er helt nødvendig i siste del av denne eksamen.

Hele oppgavesettet er arbeidskrevende, og det er ikke forventet at alle skal klare alt. Tenk strategisk i forhold til ditt nivå og løs oppgavene som er enklest for deg først.

Deloppgavene i "tematiske" oppgaver er organisert i en logisk rekkefølge, men det betyr ikke at det er en økende vanskelighetsgrad utover i deloppgavene. Hoveddelene av eksamensoppgaven teller i utgangspunktet med den andelen som er angitt i prosent, men den prosentvise uttellingen for hver oppgave kan likevel bli justert ved sensur, basert på hvordan oppgavene har fungert. De enkelte deloppgaver kan også bli tillagt forskjellig vekt.

Før den detaljerte evalueringen av eksamen din, vil vi foreta automatisk testing og plagiatkontroll av all koden du har levert. Basert på resultatene kan det hende vi ber deg om en forklaring, og dette kan

påvirke eksamensresultatet ditt. Du må huske å legge til kommentarlinjer i koden din, siden det også kan hjelpe oss å forstå koden din bedre.

Helt på slutten av eksamen ber vi deg laste opp en .zip-fil som inneholder alle de komplette kodefilene tilhørende den siste (største) delen av eksamen. Koden din trenger IKKE å være feilfri eller å kjøre uten problemer for at du skal få bestått, men det er en fordel å sende inn kjørende kode. Vi vil evaluere hvert spørsmål i detalj for å se hvor mye du har lært av dette kurset. For å få bestått på denne eksamen er det HELT AVGJØRENDE AT DU LASTER OPP ZIP-FILEN, innen de 4 timene til rådighet. Prøv å last opp filen en gang midt i eksamenstiden, for å se at du klarer det, og hvor lang tid du bruker på det.

0.2 Eksamen - Step by Step

Denne listen vil lede deg trinn for trinn igjennom hva du skal gjøre på denne eksamen.

1. Les nøye igjennom disse **introduksjons-sidene**.
2. **Last ned** medfølgende .zip-fil med en gang eksamen starter. I .zip-filen finner du en mappe med .cpp- og .h-filer, og en undermappe med datafiler.
3. Les **forklaringen** på problemet gitt i begynnelsen av hver seksjon.
4. I .h og .cpp-filene finner du både **grunnleggende kode** og **oppgavene** du må fullføre. Oppgavene står også i del 3 på Inspera.
5. Du kan bruke VS Code (eller et hvilket som helst annet utviklingsmiljø du foretrekker) til å åpne og jobbe med den oppgitte koden (som i Øving 0).
6. Du kan bruke boken eller andre online/offline ressurser, men du kan **IKKE** samarbeide med andre på noen måte, eller direkte kopiere og lime inn online kode som om det er din egen.
7. Etter å ha fullført hver enkelt kodeoppgave, bør du sende inn svaret ditt gjennom Inspera.
    - Last opp all den komplette koden som en .zip-fil. Ikke endre den opprinnelige mappestrukturen. For å få bestått på denne eksamen er det HELT AVGJØRENDE AT DU LASTER OPP ZIP-FILEN, innen de 4 timene til rådighet.
    - Det er mulig å oppdatere både enkelt-svarene og filopplastningen flere ganger, i tilfelle du retter på noe etter første innlevering.
    - Prøv å last opp filen en gang midt i eksamenstiden, for å se at du klarer det, og hvor lang tid du bruker på det.
8. Send inn koden din selv om den ikke kan kompileres og / eller ikke fungerer riktig. Fungerende kode er **IKKE** et krav for at du skal stå, men det er en fordel.

0.3 Nedlasting av start-fil

Vi gir dere en .zip-fil med noen forhånds-programmerte deler av programmene og en datafil som du trenger å bruke i programmet ditt.

- Last ned og lagre .zip-filen (lenken er på Inspera). Gjør dette med en gang eksamen starter... i tilfelle noe er galt, eller internett detter ut av og til underveis.
- Husk å lagre den på et sted på datamaskinen du husker og kan finne igjen.
- Pakk ut (unzip) filen. Du finner de forskjellige oppgavene i forskjellige mapper, inkludert oppgaveforklaringer som kommentarer. Du kan også finne disse forklaringene i Inspera, som forskjellige del-spørsmål.
- Begynn å jobbe med oppgavene i VS Code (eller et hvilket som helst annet utviklingsmiljø du foretrekker). Vi forventer at du vet hvordan du sammenstiller og kjører kode, slik det ble forklart i øving 0 på begynnelsen av semesteret.

- Filene vi leverer ut kompilerer til et kjørende program, men du må selv skrive resten av koden for alle oppgavene og prøve å få hver programmet til å kjøre som et eget prosjekt. Det er ikke et krav at all den innleverte koden kan kjøres, men det er en fordel.
- Etter at du er ferdig med kodingen av alle del-spørsmål, må du laste opp alle kodefilene dine, etter at de på nytt er pakket sammen til en lignende .zip-fil som den du begynte med. Ikke endre noe på de opprinnelige mappe/fil-navnene før du zipper filen og laster den opp til Inspera igjen. Last gjerne opp på nytt hver time!