

1 1.1

Hvilke (en eller flere) av følgende utsagn er korrekt?

Velg ett eller flere alternativer

- ☐ En typeløs peker (*void peker*) er det nærmeste C++ kommer en ren maskinadresse
- ☐ Hvis man trenger kode som kjøres raskt er C++ et dårlig verktøy
- ☐ *Templates* kan ikke brukes når man lager generelle klasser
- ☐ C++ støtter objektorientert programmering

Maks poeng: 5

2 1.2

Hvilke (en eller flere) av følgende utsagn er korrekt?

Velg ett eller flere alternativer

- ☐ En klasse kan ha flere konstruktører
- ☐ Medlemmer i en klasse er *public* som standard (*default*)
- ☐ En klasse kan ikke ha flere destruktører
- ☐ *auto*-typen er spesielt nyttig i generisk kode

Maks poeng: 5

3 1.3

Hvilke (en eller flere) av følgende utsagn er korrekt?

Velg ett eller flere alternativer

- ☐ *this*-pekeren til et objekt er lik pekeren til objektet
- ☐ Det er ikke mulig å endre på *this*-pekeren i en medlemsfunksjon
- ☐ En *this*-peker kan ikke legges til i en vektor
- ☐ *this*-pekeren må slettes i destruktøren

Maks poeng: 5

4 1.4 - isYellow

Kodesnutten under inneholder en eller flere feil.

```
enum class Color{Red = 0, Blue = 1, Yellow = 2};

bool isYellow(Color c) {
    return c == 2;
}
```

Hvilke (en eller flere) av alternativene under fungerer som ønsket?

Velg ett eller flere alternativer

```
enum class Color{Red = 0, Blue = 1, Yellow = 2};
```

☐

```
bool isYellow(Color c) {
    return c == Color::Yellow;
}
```

```
enum class Color{Red = 0, Blue = 1, Yellow = 2};
```

☐

```
bool isYellow(Color c) {
    return c == Yellow;
}
```

```
enum class Color{Red = 0, Blue = 1, Yellow = 2};
```

☐

```
bool isYellow(Color c) {
    return c == Color::Yellow;
}
```

```
enum class Color{Red = 0, Blue = 1, Yellow = 2};
```

☐

```
bool isYellow(Color c) {
    return c == 2;
}
```


5 1.4 - diceValues

Kodesnutten under handler om terninger (*dice*), men den inneholder en eller flere feil som gjør at koden ikke vil kjøre/ikke vil kjøre som ønsket dersom *diceValues()*-funksjonen kalles fra f.eks. *main.cpp*-filen.

```
1  //The code in the file DiceValues.h
2  #pragma once
3  void diceValues(int numberOfDice);

1  //The code in the file DiceValues.cpp
2  #include "DiceValues.h"
3  #include "std_lib_facilities.h"
4
5  vector<int> diceValues(int numberOfDice) {
6      vector<int> diceValues;
7      constexpr int maxDiceValue = 6;
8
9      for(int i = 0; i < numberOfDice; ++i) {
10         int currValue = rand() % maxDiceValue;
11         diceValues[i] = currValue;
12     }
13     return diceValues;
14 }
```

På hvilke (velg en eller flere) kodelinjer oppstår feilen(e)? Bak hvert kodelinjetall-alternativ står det en parentes som indikerer om linjen er i *headerfilen* (.h) eller i *.cpp-filen* (.cpp).

Velg ett eller flere alternativer

- ☐ 7 (.cpp)
- ☐ 13 (.cpp)
- ☐ 11 (.cpp)
- ☐ 6 (.cpp)
- ☐ 1 (.cpp)
- ☐ 2 (.h)
- ☐ 3 (.h)
- ☐ 9 (.cpp)
- ☐ 10 (.cpp)

Maks poeng: 5

6 1.5

Se koden under.

```

1  #include "std_lib_facilities.h"
2
3  class MyStack {
4  public:
5      vector<int> vec;
6
7      void push(int number);
8
9      void pop();
10 };
11
12 inline void MyStack::push(int number) {
13     vec.push_back(number);
14 }
15
16 inline void MyStack::pop() {
17     vec.pop_back();
18 }

```

Vi ønsker å gjøre stakken om fra noe som bare håndterer heltall til noe som håndterer ulike typer ved å bruke templates. Bruk nedtrekksmenyen for å velge hvordan koden da skal se ut. Blank (tomt felt i nedtrekksmenyen) er også mulig. **Men husk å aktivt velge blankt felt i nedtrekksmenyen dersom du mener det er rett svar på noen av plassene!**

inline-nøkkelordet er brukt for å ha alt i *headerfilen*.

#include "std_lib_facilities.h"

Velg alternativ (template<T>, template<const T>, template<static T>, , template<typename T>)

```
class MyStack Velg alternativ (<static T>, <typename T>, <const T>, , <T>){
public:

    vector Velg alternativ (<typename T>, <T>, , <static T>, <int>, <const T>) vec;

    void push( Velg alternativ (, template T, static T, const T, int, T) number);

    void pop();
};

Velg alternativ (template<typename T>, template<T>, template<static T>, , template<const
T>)

inline void MyStack Velg alternativ (<int>, <typename T>, , <const T>, <static T>, <T>)::push(

Velg alternativ (const T, typename T, int, T, static T, ) number) {

    vec.push_back( Velg alternativ (const T, int, T, static T, , typename T)number);
}

Velg alternativ (template<static T>, , template<typename T>, template<T>, template<const
T>)

inline void MyStack Velg alternativ (<static T>, , <typename T>, <int>, <T>, <const T>)::pop() {
    vec.pop_back();
}
```

Maks poeng: 10

7 1.6

Se på koden under.

```
#include "std_lib_facilities.h"

class Cat {
protected:
    string name;
    int age;
public:
    Cat(string name, int age):
        name{name}, age{age} {}

    virtual void meow() {cout << "Meow\n";}
    Cat& operator=(const Cat&) = delete;

    ~Cat() {}
};

class NorwegianCat : public Cat {
public:
    NorwegianCat(string name, int age = 12) :
        Cat{name, age} {}
    void meow() override {cout << "Meow (in Norwegian)\n";}
};
```

Hvilke (en eller flere) av de følge påstandene er korrekte?

Velg ett eller flere alternativer

- ☐ *NorwegianCat* har tilgang til medlemsvariablene i *Cat*-klassen
- ☐ Funksjonen *meow()* i *Cat* er virtual
- ☐ Funksjonen *meow()* i *NorwegianCat* er virtual
- ☐ *Cat* er en abstrakt klasse

Maks poeng: 10

8 1.7

Hvilke (en eller flere) av alternativene er nødvendige når det brukes dynamisk minne i en klasse?

Velg ett eller flere alternativer

- ☐ Konstruktør
- ☐ Destruktør
- ☐ Operatoroverlasting av *operator=*
- ☐ Grunn kopiering

Maks poeng: 5

9 1.8

Gitt klassedeklarasjonen for *Hometown* under.

```
class Hometown{
    string name;
public:
    Hometown(string name) : name{name} {}
};
```

I hvilke (en eller flere) av klassedeklarasjonene under kan det være lurt å implementere kopikonstruktøren selv?

Velg ett eller flere alternativer

☐

```
class Person {
    string name;
    Hometown * home = nullptr;
public:
    Person(const Person & Me) = default;
    Person(string name, string city) : name{name} {home = new Hometown(city);}
};
```

☐

```
class Person {
    string name;
    Hometown home;
public:
    Person(string name, string city) : name{name}, home{city} {}
    Person& operator=(Person&) = delete;
};
```

☐

```
class Person {
    string name;
    Hometown home;
public:
    Person(string name, string city) : name{name}, home{city} {}
    Person() = delete;
};
```

☐

```
class Person {
    string name;
    Hometown * home = nullptr;
public:
    Person(string name, string city) : name{name} {home = new Hometown(city);}
};
```

Maks poeng: 5

10 1.9

Gitt følgende kodesnutt:

```
#include "std_lib_facilities.h"

int main(){
    vector<int> myVec{2, 5, 42, 8};
    auto iter = myVec.begin();
    vector<int>* ptr = &myVec;
    return 0;
}
```

Hvilke (en eller flere) av disse påstandene stemmer?

Velg ett eller flere alternativer

- ☐ *ptr* og *iter* peker på det samme
- ☐ *(*ptr).at(2)* og *ptr->at(2)* betyr det samme
- ☐ *iter[3]* og **(myVec.end()-1)* betyr det samme
- ☐ *iter* og *myVec.rend()* peker på samme element

Maks poeng: 5

¹¹ 2.1 - konstante referanser

Hvilke fordeler er det med konstante referanser (*const reference*)? **Forklar kort.**

Skriv ditt svar her

Maks poeng: 5

12 2.1 - smartpekere

Hvilke fordeler er det med smartpekere framfor "vanlige" pekere? **Forklar kort.**

Skriv ditt svar her

Maks poeng: 5

13 2.2

Vær oppmerksom på at denne oppgaven har to deloppgaver (a og b). Spesifiser hvilken deloppgave du svarer på ved å først skrive **a)** eller **b)**.

Se på klassedeklarasjonene under:

```
1  class Node {
2      int nodeValue;
3      Node * nextNode = nullptr;
4
5  public:
6      Node(int nodeValue) : nodeValue{nodeValue} {}
7
8      void addNode(int nodeValue) {
9          nextNode = new Node(nodeValue);
10     }
11     Node * getNextNode(){return nextNode;}
12     int returnValue(){return nodeValue;}
13
14     ~Node() {
15         if(nextNode) delete nextNode;
16     }
17 };
```

a) Hvorfor risikerer koden minnelekkasje?

b) Hvordan kan problemet løses?

Skriv ditt svar her

Format ▾

↺

✎

Σ

✖

Words: 0

Maks poeng: 10

Words: 0

Maks poeng: 10

15 2.3 - Banana

Vær oppmerksom på at denne oppgaven har to deloppgaver (a og b). Spesifiser hvilken deloppgave du svarer på ved å først skrive **a)** eller **b)**.

Se på klassen under:

```

1  //This code is in the file Banana.h
2  #pragma once
3  #include "std_lib_facilities.h"
4
5  class Banana {
6      int sweetness = 70;
7      int weight = 100;
8      bool isRipe = true;
9  public:
10     friend ostream& operator<<(ostream& os, const Banana & b);
11 };

```

```

1  //This code is in the file Banana.cpp
2  #include "Banana.h"
3  ostream& Banana::operator<<(ostream& os, const Banana & b){
4      os << b.weight << " grams banana with " << b.sweetness << ".";
5      if(b.isRipe) os << " Ready to be eaten!";
6      return os;
7  }

```

a) Hva er problemet med klassen? Forklar kort.

b) Hvordan kan problemet med denne koden minimum løses? (Med andre ord: Hva er den enkleste måten å løse dette problemet på?) **Forklar kort.** Skriv forslag til kode som løser problemet, men kun hva du vil legge til/fjerne/endre og hvor i koden dette er. Du trenger ikke å skrive inn hele koden på nytt.

Skriv ditt svar her

Words: 0

Maks poeng: 10

16 2.4 - .at()

Hvorfor er det lurt å bruke vektorfunksjonen `.at()` i stedet for tabellindeksoperatoren (`operator[]`) dersom man vil aksessere et element i en vektor? **Forklar kort.**

Skriv ditt svar her

Format

 Σ 

Words: 0

Maks poeng: 5

17 2.4 - tilordningsoperator

Hvorfor kan det være nyttig å returnere en referanse til et objekt i implementasjon av tilordningsoperatoren? **Forklar kort.**

Skriv ditt svar her

Maks poeng: 5

18 2.5

Se på koden under:

```

1  #include "std_lib_facilities.h"
2
3  class Drink{
4  public:
5      virtual void serveDrink() const {cout << "Here is your drink!\n";}
6  };
7  class Tea : public Drink{
8  public:
9      void serveDrink() const override {cout << "Here is your tea!\n";}
10 };
11 class HotChocolate : public Drink{
12 public:
13     void serveDrink() const override {cout << "Here is your hot chocolate!\n";}
14 };
15
16 int main(){
17     vector<Drink*> drinks;
18
19     Tea earlGrey{};
20     HotChocolate cocoa{};
21     Drink mysteryDrink{};
22     Tea greenTea;
23
24     drinks.push_back(&earlGrey);
25     drinks.push_back(&cocoa);
26     drinks.push_back(&mysteryDrink);
27     drinks.push_back(&greenTea);
28
29     for(auto d : drinks){
30         d->serveDrink();
31     }
32 }

```

Hvorfor lønner det seg å la elementene i vektoren *drinks* i *main()* være av typen *Drink** isteden for typen *Drink*? **Forklar kort.**

Skriv ditt svar her

Words: 0

Maks poeng: 10

19 Nedlasting/opplasting av kode v23.1

LAST NED UTDELT KODE

[handout_2023.zip](#)

Se instruksjoner på forrige side

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 du har til rådighet.

Det er mulig å oppdatere både enkeltsvarene 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



