Del I: Flervalgsoppgaver (60p)

Del 1 av eksamen består av flervalgsoppgaver. Denne delen inneholder **12 oppgaver** som totalt kan gi 60 poeng og teller ca. **20 %** av eksamen.

Informasjon om poenggiving: For hver oppgave på denne delen får du en *maksimal poengsum* på **5 poeng**. Det er minst et riktig svaralternativ per oppgave, men antall riktige svaralternativ per oppgave kan være mer enn en. For hvert riktige svaralternativ du velger får du $+\frac{\textit{maksimal poengsum}}{\textit{antall riktige svaralternativ}}$ *poeng*. For hvert urikti-ge svaralternativ du velger får du $-\frac{\textit{maksimal poengsum}}{\textit{antall riktige svaralternativ}}$ *poeng*. Uansett er det slik at det ikke er mulig å få en negativ poengsum for en oppgave; du får *minimum 0 poeng* per oppgave.

Grunnleggende kunnskap

1. (5 points) Se på funksjonsdeklarasjonen under.

   ```
   void f(int* a);
   ```

   Hvilke (en eller flere) funksjonsdeklarasjoner er ekvivalente?

   ```
   void f(int a[]);
   void f(int a[2048]);
   void f(std::array<int, 2048> a);
   void f(int** a);
   ```

Variabeldeklarasjoner og datatyper

2. (5 points) Hvilke (en eller flere) av parameteroverføringene krever at en funksjon må ha en returverdi hvis man skal se en endring utenfor funksjonsskopet?

   ```
   pass-by-value
   pass-by-reference
   pass-by-const-reference
   pass-by-pointer
   ```

3. (5 points) Hvilke (en eller flere) av følgende utsagn er korrekt?

   - En konstant (const) variabel kan endres
   - En konstant (const) referansevariabel kan kun leses
   - Man kan endre hva en referansevariabel referer til så mange ganger man ønsker
   - Funksjonsparametre som er konstante referansevariabler signaliserer at funksjonen ikke kommer til å endre parameterverdiene

4. (5 points) Se på koden under.

   ```
   #include <string>
   #include <vector>
   using namespace std;

   class ShoppingCart {
       vector<string> items;
       vector<double> prices;
   ```

```
public:
    vector<string>& getItems() {
        return items;
    }
    vector<double>& getPrices() {
        return prices;
    }
    double computeTotal() {
        double sum = 0;
        for (int i = 0; i < prices.size(); ++i) {
            sum += prices[i];
        }
        return sum;
    }
};
```

Velg riktig alternativ fra nedtrekksmenyen for hvert utsagn.

- Prices er i (klassens skop / globalt skop / lokalt skop / skopet til for-løkken)
- sum er i (klassens skop / globalt skop, lokalt skop / skopet til for-løkken)

Typekonvertering

5. (5 points) Se på typekonverteringen i koden under.

```
float floatValue = 3.5f;
int intValue = floatValue;
```

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

- Konverteringen er eksplisitt
- Konverteringen medfører tap av presisjon
- Man kan bruke static_cast<int>(floatValue) eller int(floatValue) for å unngå tap av presisjon i konverteringen
- Man vil tape presisjon selv om man bruker static_cast<int>(floatValue) eller int(floatValue), men man burde likevel bruke dem

Templates

6. (5 points) Hvilke (en eller flere) av følgende utsagn er korrekt?

- Templates muliggjør generisk programmering i C++
- Templates blir instansiert ved kjøretid
- Polymorfisme er ikke et konsept i objektorientert programmering
- Templatees blir instansiert ved kompileringstid

Overlasting

7. (5 points) Hvilke (en eller flere) av følgende utsagn er korrekt?

- Konstruktører kan overlastes
- Overlastning kan være et alternativ til å bruke standard (default) argument

- En medlemsfunksjon som er definert med virtual kan ikke overlastes
- Det er valgfritt å overlaste en medlemsfunksjon som er definert med override

Minne

8. (5 points) Koden under er et eksempel på...

```
for (int i = 0; i < 3; i++) {
        int* values = new int[3] {1, 2, 3};
}
```

- Minnelekkasje
- Hengende referanse (dangling reference)
- Dobbel frigjøring (double free)
- Dereferering av nullpeker

9. (5 points) Hvilke (en eller flere) av funksjonene kan føre til at vektorens minneavtrykk øker?

- `size()`
- `resize()`
- `reserve()`
- `push_back()`

Klasser og arv

10. (5 points) Hvilke (en eller flere) av de følgene klassedeklarasjonene har en standard (default) konstruktør?

```
class Class {
public:
    int number;
    Class(int number) : number{number} {}
};
```

```
class Class {
    int number;
public:
    Class() = default;
};
```

```
class Class {
    int number;
public:
    Class(const Class&) = delete;
};
```

```
class Class {
    int number;
};
```

11. (5 points) Se på koden under.

```
#include <set>
using namespace std;

class ImmutableSet {
    set<int> elements;
public:
    ImmutableSet() = default;
    ImmutableSet(initializer_list<int> l) : elements{l} {}
    bool contains(int val) {return elements.find(val) != elements.end();}
    friend ImmutableSet operator+(const ImmutableSet &lhs, const ImmutableSet &rhs);
};

ImmutableSet operator+(const ImmutableSet &lhs, const ImmutableSet &rhs) {
    ImmutableSet result;
    result.elements.insert(lhs.elements.begin(), lhs.elements.end());
    result.elements.insert(rhs.elements.begin(), rhs.elements.end());
    return result;
}
```

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

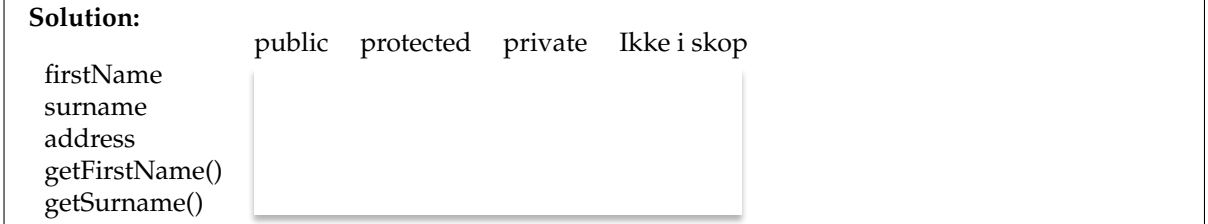
- Hvis ImmutableSet::elements-variabelen hadde vært public, hadde friend vært unødvendig i ImmutableSet-klassen
- Koden kopilerer ikke fordi ImmutableSet::elements er private
- Hvis operator+ hadde vært en medlemsoperator ville friend vært unødvendig i ImmutableSet-klassen
- Det går ikke an å lage tomme ImmutableSet-objekter

12. (5 points) Se på deklarasjonene til klassene `Parent` og `Child` under.

```
1  class Parent {
2  private:
3      string firstName;
4  protected:
5      string surname;
6      string address;
7  public:
8      string getFirstName() { return firstName; }
9      string getSurname() { return surname; }
10 };
11
12 class Child: public Parent {
13 private:
14     string secret;
15 };
```

Hva er tilgangsnivået til medlemsfunksjonene og medlemsvariablene i `Child`-klassen? (Kryss av i riktig boks for hver variabel eller funksjon.)

Solution:


public
protected
private
Ikke i skop
firstName
surname
address
getFirstName()
getSurname()

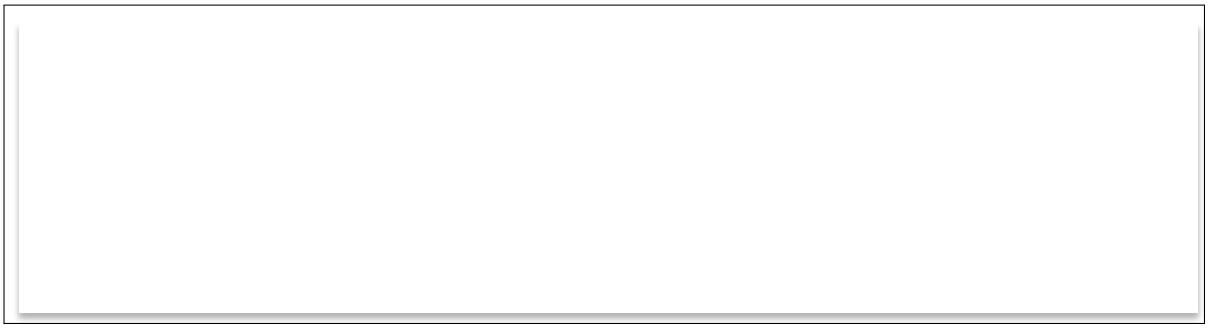
Del II: Kortsvarsoppgaver (60p)

Del 2 av eksamen er kortsvarsoppgaver. Du skal svare på oppgavene i konteksten av programmeringsspråket C++ ut i fra det du har lært i emnet gjennom forelesningene, øvingene og pensumboken *Programming: Principles and Practice Using C++, 2nd edition* av Bjarne Stroustrup. Denne delen av eksamen inneholder **10 oppgaver** som til sammen gir en maksimal poengsum på 60 poeng og teller ca. **20 %** av eksamen.

Informasjon om poenggiving: Hver oppgave gir *maksimal poengsum* på **5 eller 10 poeng** avhengig av vanskelighetsgrad og arbeidsmengde. Du får poeng etter hvor presist svaret ditt er. Det vil si at det er mer uttelling for et kort og presist svar enn et langt og upresist svar.

Grunnleggende kunnskap

1. (5 points) Forklar kort hovedforskjellen mellom *prosedyreorientert programmering* og *objektorientert programmering*.



2. (5 points) Forklar kort hvilke fordeler det er med *objektorientert programmering*.

Variabeldeklarasjoner og datatyper

3. (5 points) Forklar kort i hvilke ulike tilfeller man bør bruke de ulike parameteroverføringene `pass-by-value`, `pass-by-reference` og `pass-by-const-reference`.

Inn/ut datahåndtering

4. (5 points) Forklar kort hvordan inn/ut datahåndtering (`input/output handling`) foregår med strømmodellen (`I/O stream model`).

Kontrollstrukturer

5. (5 points) Konverter følgende `while`-løkke til en `for`-løkke.

```
#include <iostream>
using namespace std;

int a = 1;
int b;
while (a <= b){
    b++;
    cout << a << endl;
    a *= 2;
}
```

Solution:

Grafisk brukergrensesnitt

6. (5 points) Forklar kort fremgangsmåten for å opprette et *grafisk brukergrensesnitt*.

Feil

7. (5 points) Forklar kort hovedforskjellen mellom *kompileringsfeil*, *kjøretidsfeil* og *logiske feil*.

Egendefinerte typer

8. (10 points) Se på koden under.

```
#include <iostream>
using namespace std;

enum class PaintColour {Red, Yellow, Blue};

void printNumber(PaintColour colour) {
    cout << static_cast<int>(colour) << endl;

}

int main() {
    PaintColour paint = 1;
    printNumber(paint);
    return 0;
}
```

   (a) (5 points) Hvorfor kompilerer ikke koden? Forklar kort.

   (b) (5 points) Hva må endres for å løse problemet?

Klasser og arv

9. (5 points) Forklar kort i hvilke ulike tilfeller man bør bruke de ulike tilgangsnivåene `public`, `private` og `protected` når man implementerer en klasse.

10. (10 points) Se på koden under.

```
using namespace std;

class A {
int numberOfA;

public:
    A(int numberOfA=0) :
    numberOfA{numberOfA} {}
    A(const A& a);
};

class B : public A {
    int numberOfB;

public:
    B(int numberOfB=0) :
    numberOfB{numberOfB} {}
};


int main() {
    B b(5);
    B b2 = b;
    return 0;
}
```

(a) (5 points) Hvorfor kompilerer ikke koden? Forklar kort.

(b) (5 points) Hva må endres for å løse problemet? Forklar kort hvilke kodelinjer som må endres gitt at man ikke skal endre noe i `main`.