Løsninger til OOP reeksamen E 2016, 21/3-2017

```
// Opgave 1 A

// MoviePerson.h
#pragma once
#include <string>
#include <iostream>
using namespace std;

class MoviePerson
{
public:
		MoviePerson(string name);

		void setName(string);
		string getName() const;

		MoviePerson &addOscar();
		int getNumberOfOscars() const;

		void print() const;

private:
		string name_;
		int numberOfOscars_;
};

// Movieperson.cpp
#include "MoviePerson.h"
#include <iostream>


MoviePerson::MoviePerson(string name)
{
		name_ = name;
		numberOfOscars_ = 0;
}

void MoviePerson::setName(string name)
{
		name_ = name;
}

string MoviePerson::getName() const
{
		return name_;
}

MoviePerson& MoviePerson::addOscar()
{
		numberOfOscars_++;

		return *this;
}

int MoviePerson::getNumberOfOscars() const
{
		return numberOfOscars_;
}

void MoviePerson::print() const
{
		cout << getName() << ", " << getNumberOfOscars() << " Oscars" << endl;
}
```

```
// Opgave 1B
#include "MoviePerson.h"
#include "Movie.h"

int main()
{
		MoviePerson d("Dustin Hoffman");
		MoviePerson l("Lana Wachowski");
		MoviePerson s("Sandra Bullock");

		d.addOscar().addOscar();

		s.addOscar();

		d.print();
		l.print();
		s.print();

		return 0;
}
```

Milestone 1

```
Dustin Hoffman, 2 Oscars
Lana Wachowski, 0 Oscars
Sandra Bullock, 1 Oscars
```

```cpp
// Opgave 1 C

// Movie.h
#pragma once

#include <string>
#include "MoviePerson.h"
#include <vector>
#include <iostream>

class Movie
{
public:
		Movie(string, MoviePerson *);
		void hireActor(MoviePerson *);
		void print() const;

private:
		string title_;
		MoviePerson *directorPtr_;
		vector<MoviePerson *> actorPtrs_;
};

// Movie.cpp
#include "Movie.h"
#include <iostream>


// Initialisering af actorPtrs ikke nødvendig, men OK
Movie::Movie(string title, MoviePerson * director)
		: title_(title), directorPtr_(director), actorPtrs_()
{
}

void Movie::hireActor(MoviePerson * actorPtr)
{
		actorPtrs_.push_back(actorPtr);
}

void Movie::print() const
{
		cout << "The movie: " << title_ << endl;
		cout << "Directed by: ";
		directorPtr_->print();

		cout << "Starring:" << endl;

		// Alle variationer af for range og iterator/const_iterator er gyldige.

		for (auto a : actorPtrs_)
		{
			cout << "    ";
			a->print();
		}

		for (auto it = actorPtrs_.cbegin(); it != actorPtrs_.cend(); ++it)
		{
			cout << "    ";
			(*it)->print();
		}

		for (vector<MoviePerson *>::const_iterator it = actorPtrs_.begin(); it != actorPtrs_.end(); ++it)
		{
			cout << "    ";
			(*it)->print();
		}
}
```

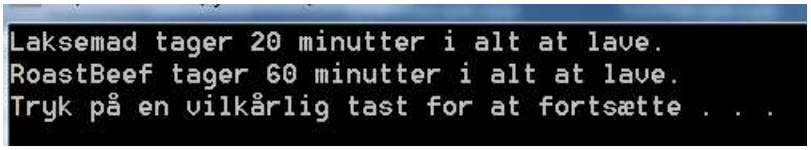
```
// Opgave 1.D

#include "MoviePerson.h"
#include "Movie.h"

int main()
{
		MoviePerson d("Dustin Hoffman");
		MoviePerson l("Lana Wachowski");
		MoviePerson s("Sandra Bullock");

		d.addOscar().addOscar();

		s.addOscar();

		d.print();
		l.print();
		s.print();

		Movie m("Matrix Revisited", &l);

		m.hireActor(&d);
		m.hireActor(&s);

		m.print();

		return 0;
}
```

Milestone 2

```
The movie: Matrix Revisited
Directed by: Lana Wachowski, 0 Oscars
Starring:
    Dustin Hoffman, 2 Oscars
    Sandra Bullock, 1 Oscars
    Dustin Hoffman, 2 Oscars
    Sandra Bullock, 1 Oscars
    Dustin Hoffman, 2 Oscars
    Sandra Bullock, 1 Oscars
Tryk på en vilkårlig tast for at fortsætte . . .
```

```
// Opgave 2A

// Ændret SimpleRecipe.h
#pragma once
#include <string>
using namespace std;

class SimpleRecipe
{
public:
		SimpleRecipe(string name, int workTime);
		string getName();
		int getWorkTime() const;
		// Virtual skal tilføjes
		virtual int getTotalTime() const;

private:
		string name_;
		int workTime_;

};


// Ingen ændringer til SimpleRecipe.cpp

// OvenRecipe.h
#pragma once
#include "SimpleRecipe.h"
class OvenRecipe : public SimpleRecipe
{
public:
		OvenRecipe(string name, int workTime, int temp, int ovenTime);
		int getTemperature() const;
		int getOvenTime() const;
		virtual int getTotalTime() const;

private:
		int temperature_;
		int ovenTime_;

};


// OvenRecipe.cpp
#include "OvenRecipe.h"

OvenRecipe::OvenRecipe(string name, int workTime, int temp, int ovenTime)
		:
		SimpleRecipe(name, workTime), temperature_(temp), ovenTime_(ovenTime)
{
}

int OvenRecipe::getTemperature() const
{
		return temperature_;
}

int OvenRecipe::getOvenTime() const
{
		return ovenTime_;
}

int OvenRecipe::getTotalTime() const
{
		return getWorkTime() + ovenTime_;
}
```

Milestone 3



```
// Opgave 3A

// Sekvens.h
#pragma once
class Sekvens
{
public:
		Sekvens();
		~Sekvens();
		Sekvens &push_back(int);
		int getLength() const;
		int getByIndex(int) const;
		void print() const;

private:
		int length_;
		int * dataPtr_;
};

// Sekvens.cpp
#include "Sekvens.h"
#include <iostream>
using namespace std;

Sekvens::Sekvens()
{
		length_ = 0;
		dataPtr_ = nullptr;
}


Sekvens::~Sekvens()
{
		delete dataPtr_;
}

Sekvens & Sekvens::push_back(int data)
{
		int * temp = dataPtr_;
		dataPtr_ = new int[length_ + 1];

		for (int i = 0; i < length_; i++)
		{
			dataPtr_[i] = temp[i];
		}
		dataPtr_[length_] = data;

		delete temp;

		length_++;

		return *this;
}

int Sekvens::getLength() const
{
		return length_;
}

int Sekvens::getByIndex(int index) const
{
		if (0 <= index && index < length_)
		{
			return dataPtr_[index];
		}
		else
		{
			return 0;
		}
}

void Sekvens::print() const
{
		for (int i = 0; i < length_; ++i)
		{
			cout << dataPtr_[i] << " ";
		}
}
```

```
// Opgave 3B

// Testprogram
#include "Sekvens.h"
#include <iostream>
using namespace std;

int main()
{
		Sekvens s;

		s.push_back(1).push_back(2).push_back(3).push_back(4).push_back(5);
		s.push_back(5).push_back(4).push_back(3).push_back(2).push_back(1);

		s.print(); cout << endl;

		cout << "Element med index 3 er " << s.getByIndex(3) << endl;


		return 0;
}
```

Milestone 4

```
1 2 3 4 5 5 4 3 2 1
Element med index 3 er 4
Tryk på en vilkårlig tast for at fortsætte . . .
```

```
// Opgave 3C+D

// Tilføjelser til Sekvens.h

// New methods for rule of 3
Sekvens(const Sekvens &);
const Sekvens& operator=(const Sekvens &);

// Tilføjelser til Sekvens.cpp

// New methods for rule of 3
Sekvens::Sekvens(const Sekvens & copyMe)
{
		length_ = copyMe.length_;
		dataPtr_ = new int[length_];

		for (int i = 0; i < length_; i++)
		{
			dataPtr_[i] = copyMe.dataPtr_[i];
		}
}

const Sekvens & Sekvens::operator=(const Sekvens & copyMe)
{
		if (this != &copyMe)
		{
			if (length_ != copyMe.length_)
			{
				delete dataPtr_;

				length_ = copyMe.length_;
				dataPtr_ = new int[length_];
			}

			for (int i = 0; i < length_; i++)
			{
				dataPtr_[i] = copyMe.dataPtr_[i];
			}
		}

		return *this;
}
```

Milestone 5

```
1 2
1 2 3 4
1 2 3 4 5 6
Tryk på en vilkårlig tast for at fortsætte . . .
```