I2-OOP sommer 2018 reeksamen – løsningsforslag

OPGAVE 1

```
// MILESTONE 1 (Kaede.h)
#pragma once

class Kaede
{
public:
	Kaede(int antalLed);
	void print() const;
	Kaede & operator+=(const Kaede &right);
private:
	int antalLed_;
	int kaedeNummer_;
	static int antalKaeder_;
};

// MILESTONE 1 (Kaede.ccp)
#include "Kaede.h"
#include <iostream>

int Kaede::antalKaeder_ = 0;

Kaede::Kaede(int antalLed)
{
	antalLed_ = (antalLed > 0 ? antalLed : 1);
	++antalKaeder_;
	kaedeNummer_ = antalKaeder_;
}

void Kaede::print() const
{
	std::cout<< "Kaede nr. " << kaedeNummer_ << " har " << antalLed_ << " led." << std::endl;
}

Kaede & Kaede::operator+=(const Kaede &right)
{
	antalLed_ += right.antalLed_;

	return *this;
}

// MILESTONE 1 (test_Kaede.ccp)
#include "Kaede.h"

int main()
{
	Kaede k1(20), k2(12);

	k1.print();
	k2.print();

	k1 += k2;

	k1.print();

	return 0;
}
```

OPGAVE 2

```
// MILESTONE 2 (Musiker.h)
#pragma once
#include <string>
#include <vector>
using std::string;

class Musiker
{
public:
	Musiker(string navn, string inst1, string inst2="", string inst3="", string inst4="");
	void print() const;
private:
	string navn_;
	std::vector<string> instrumenter_;
};

// MILESTONE 2 (Musiker.cpp)
#include "Musiker.h"
#include <iostream>

Musiker::Musiker(string navn, string inst1, string inst2, string inst3, string inst4)
{
	navn_ = navn;

	instrumenter_.push_back(inst1);

	if(inst2 != "")
		instrumenter_.push_back(inst2);
	if (inst3 != "")
		instrumenter_.push_back(inst3);
	if (inst4 != "")
		instrumenter_.push_back(inst4);
}

// MILESTONE 3 (Band.h)
#pragma once
#include"Musiker.h"
#include <string>
#include <list>

class Band
{
public:
	Band(std::string navn);
	void tilfoejMusiker(Musiker * m);
	void print() const;
private:
	std::string navn_;
	std::list<Musiker*> musikere_;
};

// MILESTONE 3 (Band.cpp)
#include "Band.h"
#include <iostream>
#include<iterator>

Band::Band(std::string navn)
{
	navn_ = navn;
}

void Band::tilfoejMusiker(Musiker * m)
{
	musikere_.push_back(m);
}
```

```cpp
void Band::print() const
{
	std::cout << navn_ << std::endl;

	std::list<Musiker*>::const_iterator i;

	for ( i = musikere_.begin(); i != musikere_.end(); i++)
		(*i)->print();
}
```

```cpp
// MILESTONE 3 (test_Band.cpp)
#include "Band.h"
#include "Musiker.h"

int main()
{
	Musikker m1("Kurt", "Sang");
	Musikker m2("Hanne", "Trommer");
	Musikker m3("Ole", "Guitar", "Bass");

	Band b("Partycrashers");

	b.tilfoejMusiker(&m1);
	b.tilfoejMusiker(&m2);
	b.tilfoejMusiker(&m3);

	b.print();

	return 0;
}
```

OPGAVE 3

```cpp
// MILESTONE 4 (Bog.h)
#pragma once

class Bog
{
public:
	Bog(const char * titel, int antalSider);
	void print() const;
	~Bog();
private:
	char * titel_;
	int antalSider_;
};
```

```cpp
// MILESTONE 4 (Bog.cpp)
#include "Bog.h"
#include <string.h>
#include <iostream>

Bog::Bog(const char * titel, int antalSider)
{
	titel_ = new char[strlen(titel) + 1];
	strcpy(titel_, titel);

	antalSider_ = antalSider > 0 ? antalSider : 0;
}

void Bog::print() const
{
	std::cout << titel_ << "   " << antalSider_ << " sider" << std::endl;
}
```

```cpp
// MILESTONE 4 (test_Bog.cpp)
#include "Bog.h"

int main()
{
	Bog b("Sort sol", 234);
	b.print();

	return 0;
}

// MILESTONE 5 (Bog.h - tilføjelser)
Bog(const Bog &copy);
~Bog();
Bog & operator=(const Bog &right);

// MILESTONE 5 (Bog.cpp - tilføjelser)
Bog::Bog(const Bog & copy)
{
	titel_ = new char[strlen(copy.titel_) + 1];
	strcpy(titel_, copy.titel_);
	antalSider_ = copy.antalSider_;
}

Bog::~Bog()
{
	delete[] titel_;
}

Bog & Bog::operator=(const Bog & right)
{
	if (this != &right)
	{
		if( strlen(titel_) != strlen(right.titel_))
		{
			delete[] titel_;
			titel_ = new char[strlen(right.titel_) + 1];
		}
		strcpy(titel_, right.titel_);

		antalSider_ = right.antalSider_;
	}
	return *this;
}
```