Opgave 1

En møntsamler og hobbyprogrammør ønsker at lave sit eget lille system til registrering af sine mønter og deres værdi. 1. iteration af systemet skal indeholde tre klasser *CoinCollection, Country* og *Coin.*

Coin er den centrale klasse, og til den kan anvendes nedenstående header-fil

```
class Coin
{
public:
            Coin();
            Coin(Country&,int, int, string, int, int);
            Country& getCountry();
            int getCatalogueNumber();
            int getYearIssued();
            string getMotive();
            int getFaceValue();
            int getValue();
            void changeValue(int); //Kan ændre møntens værdi (attributten value)
            ~Coin();
private:
            Country country; //Det land, som har udstedt mønten
            int catalogueNumber; //Katalognummer til identifikation af mønten
            int yearIssued; //Det år, mønten blev udstedt
            string motive; //Møntens motiv
            int faceValue; //Den værdi, der står på mønten i landets valuta
            int value; //Katalogværdi, dvs. den værdi mønten kan handles i
};
```

Med hensyn til værdierne, så antages det, at den pålydende værdi (face value) er i det udstedende lands valuta, mens katalogværdien er i danske kroner.

CoinCollection kan opfattes som en facade- og container-klasse, som indeholder de overordnede metoder og et array af *Coin*-objekter.

Country-klassen indeholder blot navnet på det land, mønten er udstedt i.

CoinCollection-klassen skal indeholde et array med plads til 300 *Coin*-objekter. Derudover skal der skrives to metoder:

```
int CoinCollection::getTotalValue()
```

```
int CoinCollection::getTotalValueDecade(int decade)
```

Den første metode returnerer den opsummerede katalogværdi af samtlige mønter.

Den anden metode returnerer den samlede katalogværdi for et bestemt årti angivet ved parameteren. Hvis parameteren fx er 194 returneres den samlede katalogværdi for mønter udstedt i perioden 1940 til 1949, begge år inklusive. Det antages, at parameteren er korrekt udfyldt, og der skal således ikke foretages validering.

Der skal endvidere implementeres metoder og eventuelle hjælpeattributter, så nedenstående testdriver kan eksekveres og for de to sidste metodekalds vedkommende med de angivne resultater.

```
int main()
{
		CoinCollection coinC;
		Country d("Denmark");
		Coin c1(d, 12, 1953, "Frederik 9.", 2, 23);
		coinC.addCoin(c1);
		Coin c2(d, 12, 1945, "Christian 10.", 1, 38);
		coinC.addCoin(c2);
		Coin c3(d, 12, 1965, "Frederik 9.", 5, 17);
		coinC.addCoin(c3);
		Coin c4(d, 12, 1988, "Margrethe 2.", 10, 15);
		coinC.addCoin(c4);
		Coin c5(d, 12, 1999, "Margrethe 2.", 20, 24);
		coinC.addCoin(c5);
		Coin c6(d, 12, 1948, "Frederik 9.", 1, 56);
		coinC.addCoin(c6);
		c3.changeValue(-5);
		cout << "Total value: " << coinC.getTotalValue() << endl; // 168
		cout << "Decade value 1940's: " << coinC.getTotalValueDecade(194); // 94
}
```

Opgave 2

Denne opgave går ud på at skrive en metode med følgende signatur:

```
bool neighbors(string s)
```

Der returneres *true*, hvis parameteren s indeholder nabobogstaver i 'stigende' rækkefølge, fx opfylder *gh* kriteriet; men det gør *hg* ikke. Findes der ikke nabobogstaver returneres *false*. Opgaven skal løses for små bogstaver i det engelske alfabet (`abcdefghijklmnopqrstuvwxyz`).

Eksempler:

'hsgafkroduitjsla' returnerer *false* – ingen nabobogstaver

'krstuoinnaqwlp' returnerer *true* – s og t er nabobogstaver