dokumentissa koodissa ovat kaikki tehtävät 1-4, myös bonustehtävä lisätty koodiin.

main.cpp

#include "henkilo.h" #include "paivays.h" #include "noppa.h" #include "Osoite.h" #include "Kalenterimerkinta.h" #include

```
void doSomethingArvoparametri (Kalenterimerkinta aKalenterimerkinta)
{
    aKalenterimerkinta.setAsia("ei tapahtumia");
    aKalenterimerkinta.tulostaKalenterimerkinta();
    return;
}

void doSomethingViittausParametri (Kalenterimerkinta& aKalenterimerkinta)
{
    aKalenterimerkinta.setAsia("tanaan tapahtuu vaikka mita!");
    aKalenterimerkinta.tulostaKalenterimerkinta();
}

void doSomethingViittausParametri(const Kalenterimerkinta& aKalenterimerkinta)
{
    // T�m� funktio ei voi muuttaa tuodun olion sis�ist� tilaa mitenk��n, voi
vain tulostaa jne
    // Funktio voi kutsua VAIN sellaisia metodeja, joilla ei voi muuttaa arvoja.
    //
    // aKalenterimerkinta.setAsia("tanaan tapahtuu vaikka mita!");      <--- EI
onnistu
    aKalenterimerkinta.tulostaKalenterimerkinta();                      // <---
onnistuu
}
int main() {

    /*

    Kalenterimerkinta merkinta;
    merkinta.kysyTiedot();
    merkinta.tulostaKalenterimerkinta();

    // tehdaan kopio merkinta - > nyt - olioon, jossa asetetaan uusi p�ivays ////
    ////////////////////////////////////////////////////////////////////////////
    Kalenterimerkinta nyt(merkinta);
    nyt.tulostaKalenterimerkinta();

    // vied��n merkinta-olio aliohjelmiin
    // aliohjelma kopioi koko olion aliohjelmaan, tekee sille jotain, tuhoaa sen.
```

```
P��ohjelmassa
    // pysyyy alkuper�inen olio muuttumattomana (kuluttaa muistia)
    doSomethingArvoparametri(merkinta);
    merkinta.tulostaKalenterimerkinta();

    // Kyseinen funktio muuttaa my�s p��ohjelmasta viedyn olion arvoja, jotka
tulostuvat
    // p��ohjelmassa
    doSomethingViittausParametri(merkinta);
    merkinta.tulostaKalenterimerkinta();

    */
        ////////////////////////////////////////////////////////////////////////
        //      Luodaan henkilo - vektori
        ////////////////////////////////////////////////////////////////////////

    henkilo tyyppi;
    std::vector<henkilo*> lista;
    int valikko = 0;
    bool operand = true;

    do {
        std::cout << "Valitse vaihtoehto:" << std::endl;
        std::cout << "1 = Lisaa henkilo 2 = Tulosta henkilot 3 = Poista kayttajat
4 = Poista tietty kayttaja" << std::endl;
        std::cout << "Valintasi: ";
        std::cin >> valikko;
        // Luodaan uusi kaytt�j�
        if (valikko == 1) {
            tyyppi.kysyTiedot();
            lista.push_back(new henkilo(tyyppi));
        }
        // tulostetaan vektorissa olemassa olevat k�ytt�j�t
        else if (valikko == 2) {
            for (henkilo* h : lista) {
                h->tulostaHenkilonTiedot();
            }
        }
        // tyhjennet��n vektori sek� tyhjennet��n muistiosoitteet
        else if (valikko == 3) {
            lista.clear();
            for (henkilo* h : lista) {
                delete h;
            }
        }
        // tuhotaan haluttu k�ytt�j� jos l�ytyy, sek� sen pointteri
        else if (valikko == 4) {
            std::string syote;
            int index = 0;
            std::cin.clear();
            std::cin.ignore(80, '\n');
            std::cout << "Anna poistettavan henkilon nimi: ";
            std::getline(std::cin, syote);
```

```
            for (henkilo* h : lista) {
                if (h->getNimi() == syote){
                    delete* (lista.begin() + index);
                    lista.erase(lista.begin() + index);
                    std::cout << "Henkilo poistettu" << std::endl;
                }
                index++;
            }
        }
        else
            operand = false;
    } while (operand);

    std::cout << std::endl << std::endl;
    std::cout << "Ohjelma paattynyt" << std::endl;



return 0;
}
```

henkilo.h

```
#pragma once
#include <iostream>
#include <vector>
#include <string>
#include "Osoite.h"


// Henkil? - luokan esittely (kirjoitetaan omaan .h-tiedostoon

class henkilo
{
    // luokan julkinen rajapinta (API), toiminnot, metodit
public:
    //Constuctors:
    henkilo();
    henkilo(const std::string& aNimi, const int& aIka);
    henkilo(const std::string& aNimi, const int& aIka, const Osoite& aOsoite);
    //Copy Constructor
    henkilo(const henkilo& aHenkilo);

    //Destructors:
    ~henkilo();

    void kysyTiedot();
```

```
    void setNimi(const std::string& aNimi);
    void setIka(const int& aIka);
    int getIka() const;
    std::string getNimi() const;
    void tulostaHenkilonTiedot() const;
    void kasva();

    Osoite getOsoite() const;
    void setOsoite(const Osoite& aOsoite);


    // Luokan tietoj�senet, yksityinen osuus (ylees� private)
    // N�ihin p��st��n k�siksi vain luokan metodeissa
private:

    std::string nimi;
    int ika;
    Osoite osoite;
};
```

henkilo.cpp

```
#include "henkilo.h"
#include "Osoite.h"
#include <istream>


// Henkil� - luokan m��rittely eli toteutus (implementation)


henkilo::henkilo() : nimi("---"), ika(0)
{
    std::cout << "Luodaan parametriton rakentaja" << std::endl;
}

henkilo::henkilo(const std::string& aNimi, const int& aIka) : nimi(aNimi),
ika(aIka)
{
    std::cout << "Luodaan 3-parametrinen henkilo-rakentaja" << std::endl;
}

henkilo::henkilo(const std::string& aNimi, const int& aIka, const Osoite& aOsoite)
    : nimi (aNimi), ika(aIka), osoite(aOsoite)
{
    std::cout << "henkilo-rakentaja osoite-luokan kanssa aktivoitu" << std::endl;
}
```

```
henkilo::henkilo(const henkilo& aHenkilo)
    :nimi(aHenkilo.nimi),ika(aHenkilo.ika)
{
    std::cout << "Henkilo-luokan kopiorakentaja aktivoitu" << std::endl;
}

henkilo::~henkilo()
{
    std::cout << "Henkilo-luokan purkaja aktivoitu" << std::endl;
}

void henkilo::kysyTiedot()
{
    std::cin.clear();
    std::cin.ignore(80, '\n');
    std::cout << "Tervetuloa kayttamaan ohjelmaa!" << std::endl;
    std::cout << "Anna nimi ";
    std::getline(std::cin, nimi);
    std::cout << "Anna ika: ";
    std::cin >> ika;
    std::cin.ignore(80, '\n');
    osoite.kysyOsoitetiedot();
}

void henkilo::setNimi(const std::string& aNimi) {
    // asetetaan henkil�lle nimi
    nimi = aNimi;
}

void henkilo::setIka(const int& aIka)
{
    ika = aIka;
}

int henkilo::getIka() const
{
    return ika;
}

std::string henkilo::getNimi() const
{
    return nimi;
}

void henkilo::tulostaHenkilonTiedot() const
{
    std::cout << "Henkilon tiedot" << std::endl;
    std::cout << "Nimi: " << nimi << std::endl;
    std::cout << "Ika: " << ika << std::endl;
    osoite.tulostaTiedot();
}

void henkilo::kasva()
{
```

```
    ika++;
}

Osoite henkilo::getOsoite() const
{
    return Osoite();
}

void henkilo::setOsoite(const Osoite& aOsoite)
{
    osoite = aOsoite;
}
```

kalenterimerkinta.h

```
#pragma once
#include <iostream>
#include <string>
#include <vector>
#include "paivays.h"


class Kalenterimerkinta
{
public:
    Kalenterimerkinta();
    Kalenterimerkinta(const std::string& aAsia, bool& aMuistutus, const Paivays&
aPaivays);
    //Luodaan kopiorakentaja
    Kalenterimerkinta(const Kalenterimerkinta& aKalenterimerkinta);

    ~Kalenterimerkinta();

    void kysyTiedot();
    std::string getAsia()const;
    bool getMuistutus() const;
    void setAsia(const std::string& aAsia);
    void setMuistutus(bool& aMuistutus);
    void tulostaKalenterimerkinta() const;

private:
    std::string asia;
    bool muistutus;
    Paivays paivays;
```

```
};
```

kalenterimerkinta.cpp

```
#include "Kalenterimerkinta.h"

// constuctors and destructors

Kalenterimerkinta::Kalenterimerkinta() : asia("---"), muistutus(false)
{
    std::cout << "kalenteri-luokan oletusrakentaja aktivoitu" << std::endl;
}

Kalenterimerkinta::Kalenterimerkinta(const std::string& aAsia, bool& aMuistutus,
const Paivays& aPaivays)
    :asia(aAsia), muistutus(false), paivays(aPaivays)
{
    std::cout << "Kalenteri-luokan parametrillinen rakentaja aktivoitu." <<
std::endl;
}

Kalenterimerkinta::Kalenterimerkinta(const Kalenterimerkinta& aKalenterimerkinta)
    :asia(aKalenterimerkinta.asia), muistutus(aKalenterimerkinta.muistutus)
{
    std::cout << "Kalenterimerkinta - Kopiorakentaja aktivoitu" << std::endl;
    paivays.kysyPaivays();
}

Kalenterimerkinta::~Kalenterimerkinta()
{
    std::cout << "Kalenteri-luokan purkaja aktivoitu" << std::endl;
}

void Kalenterimerkinta::kysyTiedot()
{
    std::string syote;
    std::cout <<"Anna kalenterimerkinta: ";
    std::getline(std::cin, asia);
    std::cout << "Laitetaanko muistutus? (K/E)";
    std::getline(std::cin, syote);
    if (syote == "K" || syote == "k") {
        muistutus = true;
        }
    else {
        muistutus = false;
    }
    paivays.kysyPaivays();
```

```
}

std::string Kalenterimerkinta::getAsia() const
{
    return asia;
}

bool Kalenterimerkinta::getMuistutus() const
{
    return muistutus;
}

void Kalenterimerkinta::setAsia(const std::string& aAsia)
{
    asia = aAsia;
}

void Kalenterimerkinta::setMuistutus(bool& aMuistutus)
{
    muistutus = aMuistutus;
}

void Kalenterimerkinta::tulostaKalenterimerkinta() const
{
    std::cout << "Asetit seuraavanlaisen kalenterimerkinnan: " << std::endl;
    std::cout << "Paivays: ";
    paivays.tulostaPaivaysTanaan();
    std::cout << "Asettamasi merkinta: " << asia << std::endl;
    std::cout << "Muistutus laitettu: ";
    if (muistutus)
        std::cout << "Kylla" << std::endl << std::endl;
    else
        std::cout << "Ei" << std::endl << std::endl;

}
```

Osoite.h

```
#pragma once
#include <iostream>
#include <string>
#include <vector>

class Osoite
{
public:
    Osoite();
    Osoite(const std::string& aKatuosoite, const std::string& aKunta, const
```

```
std::string& aPostinumero);
    ~Osoite();
    std::string getKatuosoite() const;
    std::string getPostinumero() const;
    std::string getKunta() const;
    void setKatuosoite(const std::string& aKatuosoite);
    void setPostinumero(const std::string& aPostinumero);
    void setKunta(const std::string& aKunta);
    void tulostaTiedot() const;
    void kysyOsoitetiedot();

private:
    std::string katuosoite;
    std::string kunta;
    std::string postinumero;
};
```

Osoite.cpp

```
#include "Osoite.h"

// constuctors and destructors

Osoite::Osoite() : kunta("N/A"), postinumero("N/A"), katuosoite("N/A")
{
    std::cout << "Osoite-luokan parametriton rakentaja aktivoitu" << std::endl;
}

Osoite::Osoite(const std::string& aKatuosoite, const std::string& aKunta, const
std::string& aPostinumero)
    : katuosoite(aKatuosoite), postinumero(aPostinumero), kunta(aKunta)
{
    std::cout << "Osoite-luokan parametrinen rakentaja aktivoitu" << std::endl;
}

Osoite::~Osoite()
{
    std::cout << "Osoite-luokan purkaja aktivoitu" << std::endl;
}

std::string Osoite::getKatuosoite() const
{
    return katuosoite;
}

std::string Osoite::getPostinumero() const
{
    return postinumero;
```

```
}

std::string Osoite::getKunta() const
{
    return kunta;
}

void Osoite::setKatuosoite(const std::string& aKatuosoite)
{
    katuosoite = aKatuosoite;
}

void Osoite::setPostinumero(const std::string& aPostinumero)
{
    postinumero = aPostinumero;
}

void Osoite::setKunta(const std::string& aKunta)
{
    kunta = aKunta;
}

void Osoite::tulostaTiedot() const
{
    std::cout << "Katuosoite on: " << katuosoite << std::endl;
    std::cout << "Postinumero on: " << postinumero << std::endl;
    std::cout << "Kunta on: " << kunta << std::endl;
}

void Osoite::kysyOsoitetiedot()
{
    std::cin.clear();
    std::cout <<"Kerro osoitteesi: ";
    std::getline(std::cin, katuosoite);
    std::cin.clear();
    std::cout << "Kerro postinumerosi: ";
    std::getline(std::cin, postinumero);
    std::cin.clear();
    std::cout << "Kerro kotikuntasi: ";
    std::getline(std::cin, kunta);
    std::cout << std::endl;
}
```

paivays.h

```
#pragma once
```

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <string>
#include <ctime>
#include <vector>


class Paivays
{
public:
    // Constructors:
    Paivays();
    Paivays(const int& aPaiva,const int& aKuukausi,const int& aVuosi);

    // Destructors:
    ~Paivays();


    void setPaiva(int& aPaiva);
    void setKuukausi(int& aKuukausi);
    void setVuosi(int& aVuosi);
    int getPaiva() const;
    int getKuukausi() const;
    int getVuosi() const;
    void tulostaPaivaysTanaan() const;
    void tulostaPaivaysHuomenna() const;
    void kysyPaivays();
    void kasvataPaivaysta();
    int tarkistaKuukausi(int& aSyote);

private:
    int paiva;
    int kuukausi;
    int vuosi;
    time_t now = time(0);
    tm* ltm = localtime(&now);
};
```

paivays.cpp

```
#include "paivays.h"

// constuctors and destructors

Paivays::Paivays() : paiva(0), kuukausi(0), vuosi(2000)
```

```
{
    std::cout << "Luodaan Paivays oletusrakentaja" << std::endl;
}

Paivays::Paivays(const int& aPaiva,const  int& aKuukausi, const int& aVuosi)
    : paiva(aPaiva), kuukausi(aKuukausi), vuosi(aVuosi)
{
    std::cout << "Luodaan 3-parametrinen rakentaja" << std::endl;
}

Paivays::~Paivays()
{
    std::cout << "Paivays-luokan purkaja aktivoitu" << std::endl;
}

// luokan metodien koodi alkaa t�st�

void Paivays::setPaiva(int& aPaiva)
{
    if (aPaiva > 0 && aPaiva <= 31)
        paiva = aPaiva;
    else {
        std::cout << "Syotit liian suuren paivamaaaran, tulostetaan oletuksena
nykyinen paivamaara." << std::endl;
        paiva = ltm->tm_mday;
    }
}

void Paivays::setKuukausi(int& aKuukausi)
{
    if (aKuukausi > 0 && aKuukausi < 13)
        kuukausi = aKuukausi;
    else {
        std::cout << "Syotit liian suuren tai pienen kuukauden, tulostetaan
nykyinen kuukausi oletuksena" << std::endl;
        kuukausi = 1 + ltm->tm_mon;
    }
}

void Paivays::setVuosi(int& aVuosi)
{
    if (aVuosi > 0 && aVuosi <= 1900 + ltm->tm_year + 10)
        vuosi = aVuosi;
    else {
        std::cout << "Syotit liian suuren vuoden (voit laittaa muistutuksen vain
10 vuoden paahan), asetetaan nykyinen vuosi" << std::endl;
        vuosi = 1900 + ltm->tm_year;
    }
}


int Paivays::getPaiva() const
{
    return paiva;
```

```
}

int Paivays::getKuukausi() const
{
    return kuukausi;
}

int Paivays::getVuosi() const
{
    return vuosi;
}


void Paivays::tulostaPaivaysTanaan() const
{
    std::cout << paiva << "." << kuukausi << "." << vuosi << std::endl;
}
void Paivays::tulostaPaivaysHuomenna() const
{
    if (kuukausi == 1 || kuukausi == 3 || kuukausi == 5 || kuukausi == 7 ||
kuukausi == 8 || kuukausi == 10 || kuukausi == 12) {
        if (paiva < 31) {
            std::cout << paiva + 1 << "." << kuukausi << "." << vuosi;
        }
        else {
            std::cout << paiva - 31 + 1 << "." << kuukausi + 1 << "." << vuosi;
        }
    }
    else if (kuukausi == 2) {
        if (paiva < 28) {
            std::cout << paiva + 1 << "." << kuukausi << "." << vuosi;
        }
        else {
            std::cout << paiva - 28 + 1 << "." << kuukausi + 1 << "." << vuosi;
        }
    }
    else {
        if (paiva < 30) {
            std::cout << paiva + 1 << "." << kuukausi << "." << vuosi;
        }
        else {
            std::cout << paiva - 30 + 1 << "." << kuukausi + 1 << "." << vuosi;
        }
    }
}

void Paivays::kysyPaivays()
{
    int pv, kk, vv;
    std::cout << "Anna paivamaara, ensin vuosi: ";
    std::cin >> vv;
    std::cout << "kuukausi: ";
    std::cin >> kk;
    std::cout << "paiva: ";
```

```
    std::cin >> pv;
    setVuosi(vv);
    setKuukausi(kk);
    setPaiva(pv);
}

void Paivays::kasvataPaivaysta()
{
    if (paiva < 31) {
        paiva++;
    }
    else {
        paiva = 1;
        kuukausi++;
    }
}

int Paivays::tarkistaKuukausi(int& aSyote)
{
    if (kuukausi == 4 || kuukausi == 6 || kuukausi == 9 || kuukausi == 11) {
        return 30;
    }
    else if (kuukausi == 2) {//helmikuu
        if (vuosi % 4 == 0) {
            return 29;
        }
        return 28;
    }
    return 31;
}
```
