Oppgaver, 16.1.17 – PG4400 C++ progr.

Oppgave 1

Skriv klassen `Die` (terning). Lag filene `Die.cpp` og `Die.h`. Den skal inneholde en `init()` funksjon som seeder random generatoren (se forelesning 1), og så skal den ha en funksjon som gir to funksjonsparametere hver et tilfeldig tall i området [1, 6]. Implementer funksjonen to ganger: en gang ved å bruke pass-by-reference og en gang ved å bruke pekere. Funksjonsprototypene skal da se ut som følger:

```
void rollDie(int& die1,  int& die2);
void rollDie(int* die1, int* die2);
```

Med tanke på god objektorientering bør du enten lage en privat hjelpefunksjon som gjør selve terningrullingen og som kalles av funksjonene over, eller implementere terningrullingen direkte i en av funksjonene over, og la den andre funksjonen være en wrapper for denne. (Med andre ord, *ikke* implementer den samme terningrulle logikken i to forskjellige funksjoner i klassen din!)

Oppgave 2 – Craps (spill)

Etter at du har implementert og testet funksjonene, kan du skrive et lite ”craps”-liknende program som lar brukeren vedde på utfallet av terningkastene. Dersom det samlede antall øyne blir 7 eller 11, vinner spilleren det dobbelte av beløpet som ble satset. Spillet pågår inntil spilleren velger å slutte eller går tom for penger. (Se wikipedia.org og søk etter ”craps” for å finne detaljerte regler for spillet dersom du vil utvide reglene).

En kjøring av programmet kan f.eks. vise en utskrift som den til høyre.

```
Hvor mye vil du spille med? 1000
Du har 1000 kroner.
Hvor mye satser du? 300
Terning 1 viser 4, terning 2 viser 5
Du tapte...
Vil du spille mer? (j, n) j
Du har 700 kroner.
Hvor mye satser du? 300
Terning 1 viser 4, terning 2 viser 2
Du tapte...
Vil du spille mer? (j, n) j
Du har 400 kroner.
Hvor mye satser du? 200
Terning 1 viser 1, terning 2 viser 2
Du tapte...
Vil du spille mer? (j, n) j
Du har 200 kroner.
Hvor mye satser du? 100
Terning 1 viser 1, terning 2 viser 3
Du tapte...
Vil du spille mer? (j, n) j
Du har 100 kroner.
Hvor mye satser du? 100
Terning 1 viser 2, terning 2 viser 2
Du tapte...
Du har ikke mer penger!
Game over!
Trykk en tast for å fortsette...
```

Oppgave 3 – Personregister

Lag klasse `Personregister` som skal være et register over personer, ved å bruke `<vector>`.
Klassen skal ha minimum følgende funksjoner:

```
void addPerson(Person);
bool deletePerson(string); // en persons personnummer er param.
string getPerson(string);  // parameter som over
int getSize();
```

Medlemsdata er i tillegg til vektoren, et heltall som skal holde rede på antall registrerte personer.
Klassen `Person` har deklarasjon som vist til høre.
Skriv en `main`-funksjon som sjekker at registeret fungerer som det skal.

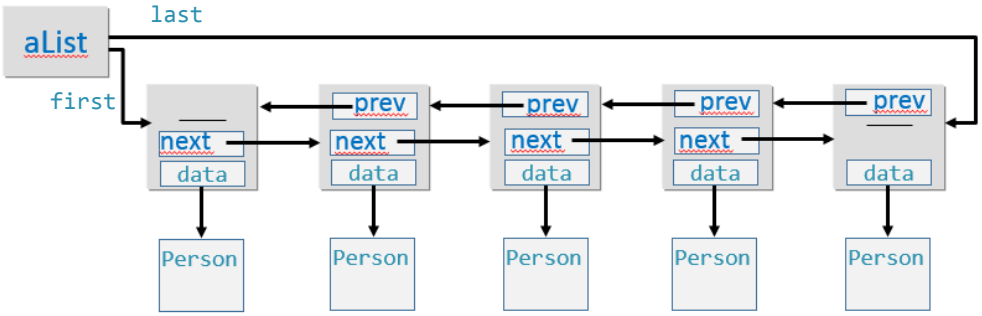
```
#include <string>

using namespace std;

class Person {
public:
    Person();
    Person(string, string, string);
    string getPNummer() const;
    string getName() const;
    string getAdress() const;
    string getData() const;
    ~Person();
private:
    string pNummer;
    string name;
    string adress;
};
```

Oppgave 4 – Lenket liste

Lag en klasse `LinkedList` som er en egen versjon av en `<list>`, som i figuren under.



`aList` er en instans av denne klassen. Den har en referanse til fremste og bakerste node i listen.
Hver node har en referanse til et `Person`-objekt, i tillegg til en referanse til neste (`next`) og foregående (`prev`) node.
Nodene er instanser av en klasse `Node` med deklarasjon som vist til høyre.
Klassen `LinkedList` skal minimum ha følgende funksjoner:

```
void addFront(Person *);
void addBack(Person *);
Person * getPerson(string);
int getSize();
void listAll();      //viser data om alle i registeret
~MyList();           //sletter alle data i registeret
```

```
#include "Person.h"

class Node {
public:
    Node(Person *);
    void setNext(Node *);
    Node * getNext();
    void setPrevious(Node *);
    Node * getPrevious();
    string getPnummer();
    Person * getData();
    ~Node();
private:
    Node * next;
    Node * previous;
    Person * data;
};
```

Klassens destruktør må sørge for at alle dynamisk opprettede objekter blir slettet når det er tid for det.
Skriv en `main`-funksjon som sjekker at registeret fungerer som det skal.