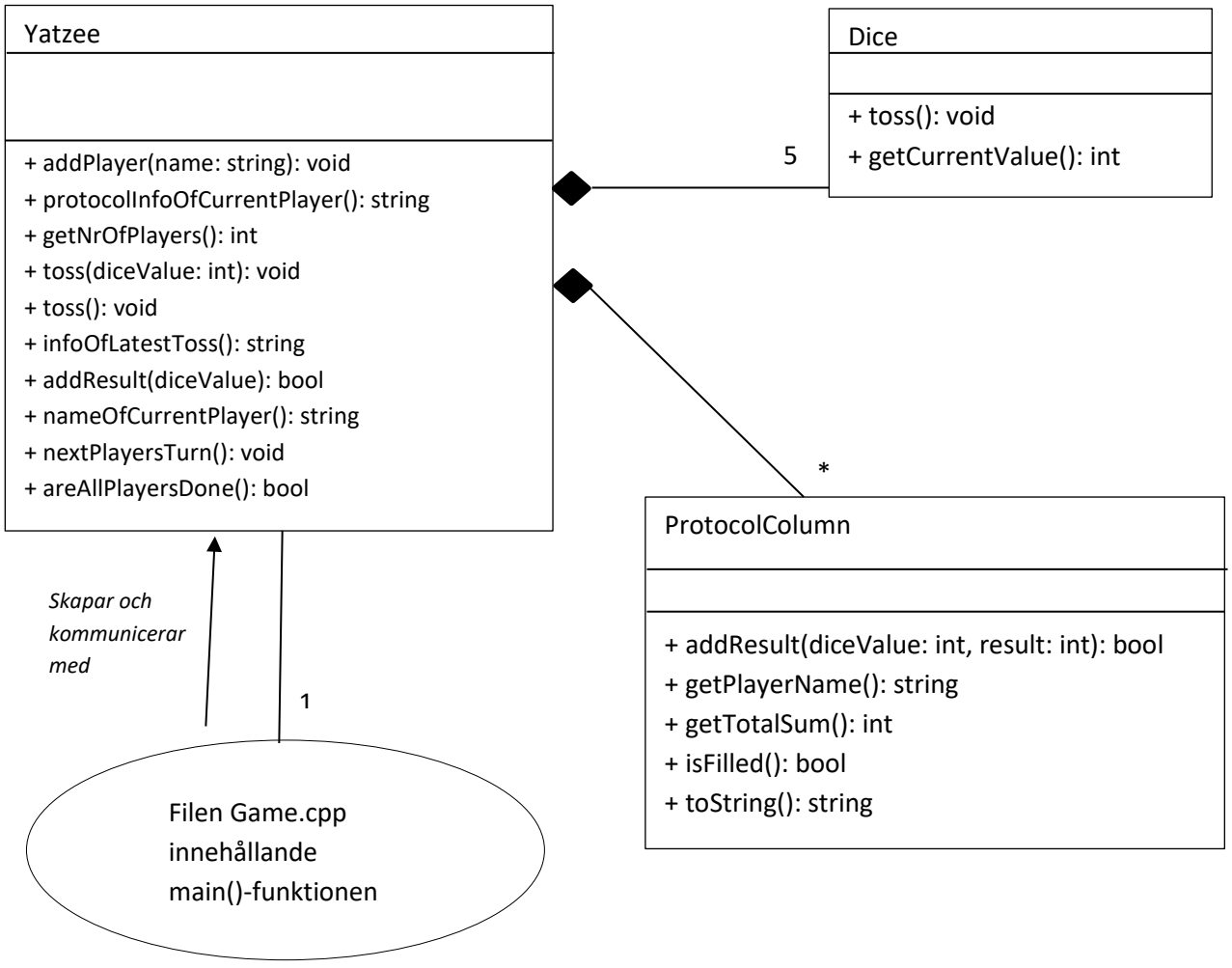
Inlämningsuppgift A

En bantad variant av tärningsspelet Yatzy

I Yatzy används 5 tärningar. I denna bantade version ska spelaren uppnå den högsta slutsumman genom att få så många ettor som möjligt, så många tvåor som möjligt osv t.o.m så många sexor som möjligt. Varje gång en spelare står i tur har denne rätt till tre tärningskast. Spelaren väljer själv vilka tärningar som skall kastas om, och poängsummanen införs i ett protokoll. Under "Fyror" ska spelaren endast skriva in poängen från de tärningar som visar fyra prickar osv.
Kontroll av om spelaren fuskar ska inte göras i denna version.
Ordningen i protokollet behöver nödvändigtvis inte följas. Exempelvis kan spelaren välja att skriva in poängsumman för ”Fyror” innan poängsumman för ”Ettor” osv.
Mer information om spelet Yatzy hittar du på exempelvis wikipedia.se.
Denna bantade version kan sägas utgöra den första översta delen av ett komplett Yatzy.

Spelet ska byggas upp av klasserna Dice, ProtocolColumn och Yatzee enligt nedan.
Du bestämmer själv lämpliga medlemsvariabler i klasserna, men relationen mellan klassen Yatzee och klassen ProtocolColumn ska implementeras med en dubbelpekare av typen ProtocolColumn (ProtocolColumn**). Du ska dessutom tillföra konstruktorer och destruktorer i alla klasser. Vidare ska du tillföra kopieringskonstruktor och tilldelningsoperator (returtyp ska vara referens) för den/de klass/klasser detta är relevant, dvs där djupkopiering krävs.



Kort beskrivning av medlemsfunktionerna:

Klassen Dice:

void toss():	slumpar fram ett nytt värde för tärningen
int getCurrentValue():	returnerar aktuellt värde för tärningen

Klassen ProtocolColumn (varje spelare har alltså en egen ProtocolColumn):

bool addResult(int diceValue,int result):	hanterar inläggning av resultat för ett givet tärningsvärde
string getPlayerName():	returnerar spelaren namn
int getTotalSum():	returnerar aktuell totalsumma
bool isFilled():	är hela protokollkolumnen ifylld?
string toString():	returnerar en sträng med all information om protokollkolumnen

Klassen Yatzee:

void addPlayer(string name):	skapar en ProtocolColumn för spelaren
string protocolInfoOfCurrentPlayer():	returnerar en sträng med all info om protokollkolumnen för aktuell spelare
int getNrOfPlayers():	returnerar antalet spelare
void toss(diceValue: int):	kastar alla tärningar som INTE har diceValue
void toss():	kastar alla tärningar
string infoOfLatestToss():	returnerar en sträng med info om alla tärningarnas värde efter det senaste kastet
bool addResult(diceValue):	summerar tärningarna med värdet diceValue och provar att lägga till resultatet till protokollkolumnen för aktuell spelare
string nameOfCurrentPlayer():	returnerar en sträng med namnet på aktuell spelare
void nextPlayersTurn():	tillser att det blir nästa spelares tur
bool areAllPlayersDone():	har alla spelare spelat klart (och fyllt i samtliga protokollkolumner)?

Du har tillgång till 3 testprogram: DiceTest.cpp, ProtocolColumnTest.cpp samt YatzeeTest.cpp. Dessa är inte fullständiga men ger dig en indikation på att du lyckats med din implementation. Börja lämpligen med att implementera klassen Dice och kör sedan testprogrammet DiceTest.cpp osv.

När testerna är gjorda ska de delar av Game.cpp som inte är klara implementeras.
Det framgår av kommentarsatser i filen vad som återstår att göra.

Generella krav:

Alla klasser ska ha privata medlemsvariabler.
Inga variabler får vara globala.
Konstanter bör vara globala.
Alla funktioner som kan vara konstanta (const) ska vara det.
Alla parametrar som bör vara konstanta (const) ska vara det.
Alla klasser delas upp i h-fil och cpp-fil.
Lösningen får inte generera några minnesläckor.
Använd `_CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF);`
i main()-funktionen för att detektera eventuella minnesläckor.