**Studentens namn: Pentikäinen, Filip**

**Generella krav**

Alla medlemsvariabler är privata: **OK**
Inga globala variabler: **OK**
Inlämning ska ske som .zip: **OK**
Alla klasser delas upp i h-fil och cpp-fil: **OK**
Ingen in- eller utmatning i medlemsfunktionerna: **OK**
Alla parametrar som bör vara konstanta (const) är det: *Ej korrekt, ingen av parametrarna för operator= och kopierings konstruktorerna är const*
Alla funktioner som kan vara konstanta (const) är det: *Ej korrekt, flertalet funktioner i register klassen som inte är const, calcVolume ska också kunna vara const*
Inga minnesläckor: **OK**
Inga tomma funktionskroppar: **OK**
Inga överflödiga medlemsvariabler: *Ej korrekt, volymen ska inte vara en variabel utan ska räknas ut av en funktion vid behov*

**Klassen Shape**

Virtuell destruktor: **OK**
Är abstrakt: **OK**
Har endast medlemsvariabel för höjd: *Ej korrekt, volym finns också*
Arv nyttjas på ett logiskt och rimligt sätt: *Ej korrekt, du ska inte ha saker som enbart rör subklasserna i basklassen (till exempel setRadius/getRadius och liknande). Dessutom så finns det ingen mening med att göra setHeight/getHeight virtuella*
Ingen onödig duplicering av kod: *Ej korrekt, båda subklasserna hanterar volymen och höjden i toString, detta kan skötas i basklassen istället, dessutom så har du duplicering av kod i dina subklassers operator= som också kan läggas i basklassen*
Virtuell toString: **OK**
Virtuell volymfunktion: *Ej korrekt, den ska returnera volymen, inte sätta en variabel*

**Klassen Cone**

Har medlemsvariabel för radie: **OK**
Överskuggad toString: **OK, men se kommentar om onödig duplicering av kod**
Överskuggad volymfunktion: *Ej korrekt, den ska returnera volymen, inte sätta en variabel*

**Klassen Box**

Har medlemsvariabler för längd och bredd: **OK**
Överskuggad toString: **OK, men se kommentar om onödig duplicering av kod**
Överskuggad volymfunktion: *Ej korrekt, den ska returnera volymen, inte sätta en variabel*

**Klassen ShapeRegister**

Kompositionen implementeras med en (1) dynamiskt allokerad array av typen Shape**: **OK**

bool addCone(egenskaper för kon): **OK**

bool addBox(egenskaper för låda): **OK**

Inga dubbletter med avseende på höjd: **OK**

Arrayen expanderar vid behov: *Ej korrekt, du ska inte behöva göra två nya allokeringar så som du gör, varför gör du inte bara den första större? Istället för att bara meningslöst föra över de till en ny array och sedan deleta den gamla och göra den större och flytta tillbaka pekarna, så behöver du bara göra en ny array som är större, flytta över alla andra pekare till den, och sedan deleta din gamla övergripande och peka om den. Det är dessutom inte särskilt effektivt att öka storleken med 1 varje gång då detta kommer leda till att många allokeringar kommer göras vilket i sin tur drar ned prestandan*

bool removeShape(höjd): *Ej korrekt, mer eller mindre allt är fel på ett eller annat sätt*

bool getAllShapesAsStrings(string arr[], int capOfArr):*Ej korrekt, du lägger på en extra sträng på arrayen som inte ska vara där och som dessutom riskerar att hamna utanför arrayens gränser*

bool getAllConesAsStrings(string arr[], int capOfArr): *Ej korrekt, du lägger på en extra sträng på arrayen som inte ska vara där och som dessutom riskerar att hamna utanför arrayens gränser*

bool getAllBoxesAsStrings(string arr[], int capOfArr): *Ej korrekt, du lägger på en extra sträng på arrayen som inte ska vara där och som dessutom riskerar att hamna utanför arrayens gränser*

capOfArr används på rätt sätt: **OK**

Returvärdet på getAll-funktionerna indikerar om ifyllningen lyckades: **OK**

bool editACone(egenskaper för kon): **OK, men egentligen så ska du casta om pekarna till sina korrekta typer och därigenom editera de, enda anledningen att du kan göra såhär är för att du inte gjort arvet rätt**

bool editABox(egenskaper för låda): **OK, men egentligen så ska du casta om pekarna till sina korrekta typer och därigenom editera de, enda anledningen att du kan göra såhär är för att du inte gjort arvet rätt**

Höjd är unik trots edit-funktionerna: *Ej korrekt, ingen koll görs om den nya höjden redan existerar*

int nrOfShapes(): **OK**

int nrOfCones(): **OK**

int nrOfBoxes(): **OK**

Kopieringskonstruktor korrekt implementerad: *Ej korrekt, du ska inte deleta, du har ju inte allokerat något minne vid det här laget. Dessutom så gör du arrayen större än vad den ska vara eftersom du allokerar plats för size + freespace*

Tilldelningsoperator korrekt implementerad: *Ej korrekt, ingen självtilldelningskoll. Dessutom så gör du arrayen större än vad den ska vara eftersom du allokerar plats för size + freespace*

Destruktor korrekt implementerad: **OK**

**Menysystem**

Minst en funktion för respektive menyalternativ: **OK**

Menyval för att lägga till: **OK**

Menyval för att ta bort: **OK**
Menyval för de olika utskrifterna: **OK**
Menyval för editering av koner och lådor: **OK**

*Du har multipla/early returns i flertalet av dina funktioner i både register klassen och i meny funktionerna, det är inget hårt krav i just denna uppgiften men i andra uppgifter samt på tentan kommer det med största sannolikhet inte tillåtas att ni har multipla/early returns.*

**Testning**
Kopieringskonstruktor: *Ej korrekt, att bara kolla höjderna mot varandra garanterar inte att resten av datan har kopierats korrekt*
Tilldelningsoperator: *Ej korrekt, att bara kolla höjderna mot varandra garanterar inte att resten av datan har kopierats korrekt*
Destruktor: *Ej korrekt, inget test utfört*