Duggainformation

Duggan har tre uppgifter som bedöms med G eller U. Vi kräver tydlig, lättfattlig och underhållsbar kod, men vi kommer *inte* vara lika petiga som i labbserien. En godkänd lösning ska fungera korrekt, använda lämpliga språkkonstruktioner samt vara fri från minnessläckor och koduppprepning.

För godkänt på duggan krävs att alla tre uppgifter bedöms med G.

Duggatid

Du har 3 timmar på dig att lösa duggan. Om du har rätt till förlängd skrivtid är skrivtiden 4 timmar. Rätt till förlängd tid beslutas av LiU centralt.

Ämnesinnehåll

Varje uppgift på duggan relaterar till en av kursens laborationer. Detta medför att god kunskap om C++-syntax och semantik från respektive laboration behövs. Mer konkret behövs god kunskap om bland annat:

- Grunder: typer, variabler, satser, std::vector, funktioner, parametrar, argument
- Klass: objekt/instans, konstruktor, inkapsling, datamedlem, medlemsfunktion, public, private
- Minneshantering: pekarmanipulation, allokering, avallokering, minnesläckor
- Arv: basklass, härledd klass, anrop av basklasskonstruktor, protected
- Polymorfi: dynamisk/statisk bindning, virtual, slicing

Hjälpmedel

Under duggan råder vanliga tentamensregler. Du får ha med följande hjälpmedel:

- En bok om C++. För boken gäller följande regler:
  - Kommentarer eller noteringar som direkt rör text och exempel på sidan i fråga får finnas i sidmarginalen.
  - Egna sidflikar för att enkelt kunna hitta t.ex. de olika kapitlen är tillåtna.
  - Inga extra ark eller lappar, lösa eller fastsatta, får finnas.
  - Ingen programkod på tomma sidor, insidan av pärmarna, försättsblad, etc.
- Ett A4-ark. Valfritt innehåll på vardera sida. Går ej ersätta med två enkelsidiga ark.
- Penna för att anteckna. Du kan be tentamensvakt om kladdpapper.

Följande får INTE tas med:

- Elektroniska prylar. Till exempel miniräknare, mobiltelefon, hörlurar, tangentbord, smartklocka etc.

Inloggning

Tentamensvakt kontrollerar anmälan vid inpassage till salen. När datorns inloggningsskärm visar "tentamensläge aktivt" kan du logga in som vanligt med ditt LiU-ID och ditt privata lösenord och börja förbereda din programmeringsmiljö. Tentamensvakten kommer under duggan gå runt för att kontrollera legitimation och hjälpmedel.

Alias för kompilering

Under duggan finns det tre alias att använda sig av för kompilering med C++17:
`w++17` Rekommenderas!
`e++17` Kompilering med alla varningar som fel.
`g++17` Kompilering utan varningar.

Frågor

Frågor ska ställas via studentklienten. Detta för att vi ska ha en historik av konversationen samt för att vi ska kunna ge samma hjälp till all studenter. Räck upp handen om klienten, tangentbordet, musen eller datorn inte verkar fungera som de ska.

Referensmanual

Det finns tillgång till valda delar av `cppreference.com`. Du måste starta webbläsaren via skrivbordsikonen "Web access" för att komma åt sidan.

Inlämning

När du har en lösning som kompilerar utan varningar, kör utan minnesläckor och löser uppgiften skickar du in den via studentklienten. Din inlämning bedöms direkt[1] och du får svar via studentklienten. Om du får betyg G kan då gå vidare till nästa uppgift. Om du får komplettering läser du noga vad som inte fungerar, fixar till det efter bästa förmåga, och skickar in igen.

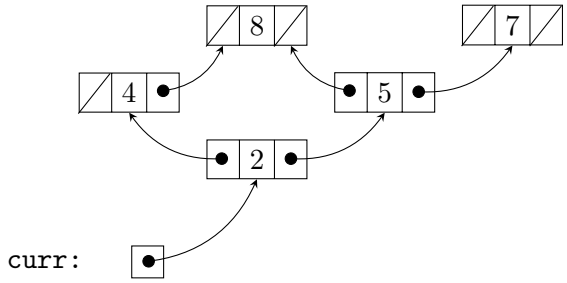
Utloggning

När du är nöjd med ditt betyg (som står i studentklienten) kan du avsluta duggan. Stäng alla program och spara alla filer. Sedan loggar du ut som vanligt i menyn. Lämna inte din plats innan inloggningsskärmen syns.

[1] Beroende på mängd samtidiga inskickningar kan det ta flera minuter att få svar, räck upp handen om du inte fått svar på 15 minuter.

Uppgift 1

Givet i `uppg1.cc` är kod som skapar följande länkade stuktur. Skriv kod som avallokerar hela strukturen.



KRAV:

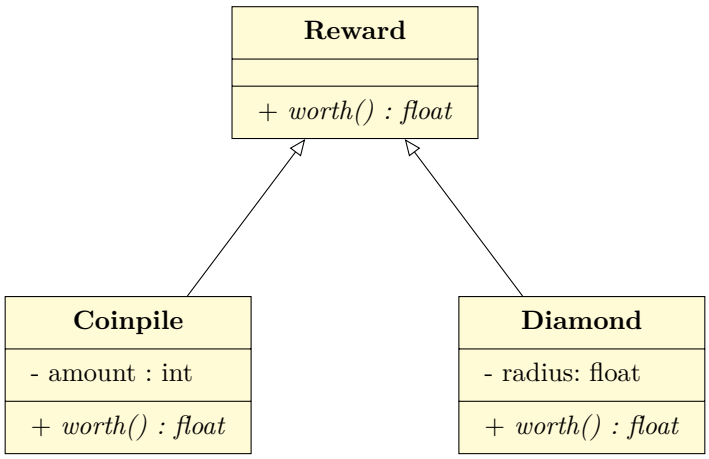
- Du måste använda den givna `Node` från `uppg1.cc`. Du får inte ändra på den givna koden, endast lägga till mer.
- Samtliga element är dynamiskt allokerade. Din kod får inte generera minnesläckor eller minnesfel.
- Du får inte deklarera några fler variabler än de som redan är deklarerade i `main()`.

TIPS: Förväntad lösning är 5-6 rader kod.

TIPS: Använd valgrind för att testa din lösning: `valgrind ./a.out`

Uppgift 2

Implementera följande klasshierarki:



- `Reward::worth()` har inget definierat beteende.
- `Coinpile::worth()` returnerar `amount * 13`
- `Diamond::worth()` returnerar `4.0 / 3.0 * 3.14 * radius * radius * radius`

KRAV:

- Alla dina klasser ska vara deklarerade och definierade i *en* källkodsfil.
- Den givna filen `uppg2.cc` har ett main-program som ska fungera oförändrat.

Funktionella krav: Körning av programmet ska ge följande utdata i terminalen:

```
Total worth from your three rewards: 294.735
```

TIPS: Förväntad lösning är ca 35 rader kod.

Uppgift 3

Simon projektleder nästa generations familjespel “Simon’s House”. I datorspelet lär och guidar spelaren individerna i en familj att ta sig igenom familjelivets toppar och dalar. En liten del i spelet är - naturligtvis - att hantera tvätten. Till detta ska det gå att köpa in en liten tvättmaskin till huset. Det är du som ombeds skriva den kod som behövs för tvättmaskinen.

En `WashMachine` ska ha följande funktionalitet:

- När den installeras på plats börjar den helt tom.
- Tvättmaskinen ska gå att fylla på med klädesplagg. Ett klädesplagg representeras av ett aggregat `Cloth` beskrivet nedan. För att hålla reda på alla klädesplagg används en std::vector (du får istället använda en std::stack om du vill).
- Tvättmaskinen ska kunna startas. När tvättmaskinen startas kommer alla klädesplagg att trasslas ihop till ett enda plagg. Detta utförs genom att skapa ett nytt plagg där namnet är sammanslagningen av alla instoppade plaggs namn, och storleken är summan av alla instoppade plaggs storlek. Namnen ska sättas samman i omvänd insättningsordning. Du ska alltså ta bort från slutet av vektorn tills den är tom och slå ihop namnen vartefter. Till slut läggs det nya trasselplagget till i vektorn (som till slut kommer ha trasslet som enda innehåll).
- Det ska gå att kontrollera om tvättmaskinen är tom.
- Det ska gå att plocka ut ett klädesplagg från tvättmaskinen. Då erhålls det plagg som senast lades in i tvättmaskinen.

Ett klädesplagg (`Cloth`) består av ett *namn* och en *storlek*. Detta kan representeras med ett enkelt aggregat (`struct`) utan inkapsling. Ett klädesplagg ska gå att mata ut på en valfri utström (`ostream&`) med `<<`-operatorn. Om du vill och finner det användbart får du även göra en operator för att slå samman två plagg.

I `uppg3.cc` finns ett givet testprogram som testar klädesplaggens utmatningsoperator och skapar och använder tvättmaskinen. All kod ska skrivas i den filen utan krav på indelning i h-fil och cc-fil. Om du är van vid den indelningen skriver du det som skulle varit i h-filen högst upp och därefter det som skulle varit i cc-filen.

Körexempel: Körning av programmet ska ge följande utdata i terminalen:

```
=== Test 0 ===
PJ: 5, JohnLongs: 10
=== Test 1 ===
T-Shirt: 8
Shoe: 8
Bed Sheet: 90
LongJohns: 10
Sock: 3
PJ: 7
=== Test 2 ===
T-ShirtShoeBed SheetLongJohnsSockPJ: 126
```