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 labserien. En godkänd lösning ska fungera korrekt, använda lämpliga språkkonstruktioner samt vara fri från minnessläckor och koduppрepning.

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, minnesläcka
- Arv: basklass, härledd klass, anrop av basklasskonstruktor, protected
- Polymorfi: dynamisk/statisk bindning, virtual, slicing

Hjälpmedel

Under duggan råder standard 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.

Inlogging

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 olika studenter. Räck upp handen om klienten, tangentbordet, musen eller datorn inte verkar fungera som de ska.

C++ referens

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 är det bara att spara, avsluta, logga ut, och framåt kvällen fira! Om du får komplettering läser du noga vad som inte fungerar, fixar till det efter bästa förmåga, och skickar in igen.

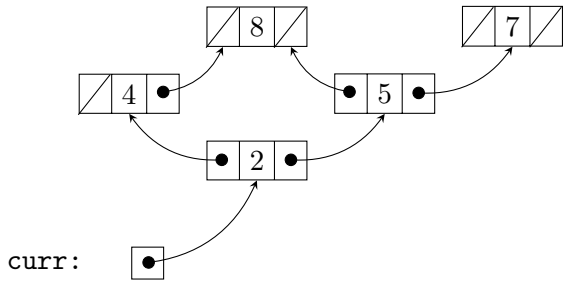
Utloggning

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 vanliga 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

Skriv kod som skapar den länkade strukturen nedan.



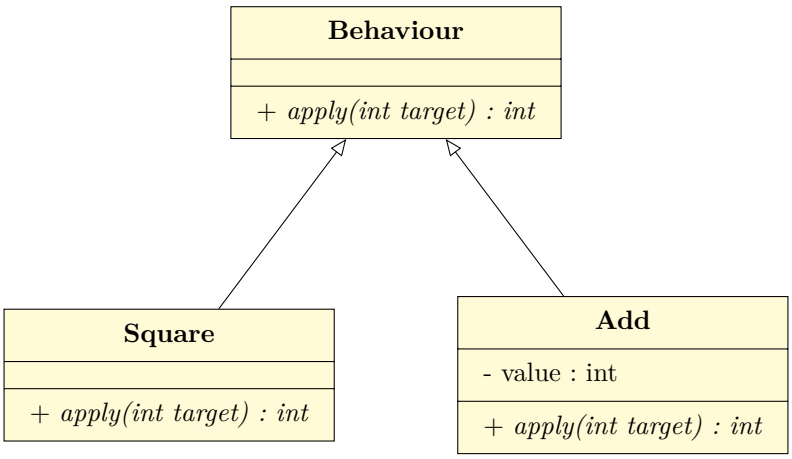
KRAV:

- Du måste använda den givna `Node` från `uppg1.cc`.
- Samtliga element ska vara dynamiskt allokerade. Du behöver inte avallokera elementen.
- Du får inte deklarera några fler variabler än de som redan är deklarerade i `main()`.

TIPS: Förväntad lösning är 2 - 7 rader kod.

Uppgift 2

Implementera följande klasshierarki:



- `Behaviour::apply(int target)` har inget definierat beteende.
- `Square::apply(int target)` returnerar `target * target`
- `Add::apply(int target)` returnerar summan av `target` och datamedlemmen `value`

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:

```
100
```

TIPS: Förväntad lösning är ca 35 rader kod.

Uppgift 3

Ett lagerhus förvarar varor som ska till ett antal butiker. En lastbil utgår från lagerhuset och kör ut till alla butiker. För att hantera butiksdestinationerna och hitta den optimala vägen behöver lastbilen en klass som håller ordning på alla destinationer.

Din uppgift är att implementera två datatyper enligt instruktionerna nedan.

Destination är ett aggregat (`struct`) som sparar ett namn på en plats samt det positiva avståndet dit.

Route_Planner är en klass som ger en lastbilschafför information om de destinationer hen ska besöka. Klassen behöver lagra

- en lista med destinationer. Listan representerar de butiker som ska besökas.

Med ett objekt av klassen ska man kunna

- beräkna den totala distansen en lastbil skulle behöva köra om den åkte fram och tillbaka från lagerhuset till varje destination i listan.
- lägga till en till destination i listan. Detta ska även gå att göra med `operator+=`.
- skriva ut alla destinationer i listan på given utström. Det ska också fungera använda `operator<<` för utskriften.

Körexempel

```
=== Alla butiker ===
# Total distans: 230
willys: 100
coop: 3
hemköp: 4
lidl: 1
ica: 7
```