Tentamensregler

Hjälpmedel

Följande får tas med på tentan:

- En bok om c++. För boken gäller följande regler:
  - Kommentarer/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.
  - Tomma sidor, insidan av pärmarna, försättsblad, etc., får inte innehålla programkod.
- Ett A4-ark med valfritt innehåll på vardera sidor (går ej att ersätta med två enkelsidiga ark).
- Penna för att anteckna under tentan. Ni kommer förses med blanka papper

Följande får INTE tas med:

- Elektroniska prylar. Dit hör till exempel miniräknare, mobiltelefon, hörlurar, tangentbord, smartklocka etc.

Utloggning

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

Bedömning

Tentan har fyra praktiska uppgifter. Två praktiska uppgifter är på grundnivå. Där visar du att du uppnår kraven för godkänt. De övriga två praktiska uppgifterna är på djupare nivå. I dessa kan du visa att du nått upp till kraven för ett högre betyg.

För betyg 3 på tentan krävs lösning av en uppgift på grundnivå. För betyg 4 krävs dessutom lösning av en påbyggnadsuppgift och för betyg 5 krävs en grunduppgift och två påbyggnadsuppgifter.

För att en uppgift ska anses godkänd krävs följande:

- att du noga följt alla instruktioner och krav ställda i uppgiften
- din kod följer god programmeringsstil (se labseriens rättningsguide)
- att klasser har ett tydligt ansvar och funktioner har en väl definierad uppgift
- att din kod har bra inkapsling och resurshantering

Bonus från labserien

Om du har bonus från labserien minskas antalet påbyggnadsuppgifter som krävs med ett, dvs för betyg 4 krävs endast en grunduppgift och för betyg 5 krävs en av varje. Bonusen är endast giltig det år den erhölls.

Frågor om uppgifter

Frågor om tentan i stort eller uppgiftspecifika frågor ska ställas via tenta-klienten. Detta för att vi ska ha en historik av konversationen samt för att vi ska kunna ge samma hjälp till olika studenter.

Systemfrågor

Om du har systemfrågor som t.ex. problem med tentaklienten eller terminalen räcker du upp handen så kommer en assistent och hjälper till.

C++ referens

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

Alias för kompilering

Under tentan 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.

Uppgift 1 - Piratkartor - Grundnivå

Du ska implementera en klass som representerar en piratkarta. Piratkartan har bredd och höjd, samt består av ett antal öar. Varje ö har ett koordinatpar och en kustlinje. Kustlinjen representeras alltid av tecknet `O`. Piratkartan implementerar:

- funktionalitet för att lägga till öar på kartan.
- funktionalitet för att kontrollera om det finns en ö på givna koordinater.
- funktionalitet som skriver ut en given x,y koordinat på angiven utström.
- utskriftsoperator som skriver ut hela kartan på angiven utström.
- additionsoperator som skapar en ny karta med alla öar från båda ursprungliga kartor. Bredd respektive höjd blir max av ursprungliga kartornas bredd respektive höjd.

Körexempel (endast sammanslagna kartan visas här):

```
$ ./a.out
+~~~~~~~~~~~~~~~~~~~~~+
|                     |
|   O                 |
|                     |
|O                    |
|                     |
|                     |
|        O            |
|                 O   |
|         O    O      |
| O   O               |
|                     |
|                     |
|                     |
|                     |
|                     |
|      O              |
|                     |
|                     |
|                     |
|   O                 |
+~~~~~~~~~~~~~~~~~~~~~+
```

Krav: Utöver bedömningskraven på sida 1 gäller att du inte får använda nyckelordet `friend`, koden ska vara korrekt uppdelad i deklarations (*.h) och implementations (*.cc) fil, implementationen ska nyttja de funktioner som finns i den mån det är lämpligt, och du ska identifiera och göra privat den funktion som är minst användbar utanför klassen. Huvudprogrammet är givet i `uppg1.cc`.

Uppgift 2 - Stilig terminal - Grundnivå

I denna uppgift ska du implementera ett system som låter oss skapa strängar med olika stilar som syns i terminalen. För att göra detta behövs följande klasser:

- Styled_String - En basklass som lagrar den ursprungliga strängen och deklarerar en funktion `get_styled()` som returnerar den stylade strängen. Basklassens version returnerar strängen rakt av (utan stil).
- Bold_String - En klass som ärver från Styled_String och implementerar funktionen `get_styled()`. Returnerar ursprungliga strängen med prefixet som ger fetstil och suffixet som återställer till normal text.
- Sparse_String - En klass som ärver från Styled_String och implementerar funktionen `get_styled()`. Returnerar ursprungliga strängen med ett blanktecken mellan varje tecken (ger gles text).

Krav: Utöver bedömningskraven på sida 1 behöver din lösning korrigera för de minnesläckor som uppstår i huvudprogrammet i `uppg2.cc`.

Körexempel:

```
$ ./a.out

Denna text är fetstil. Fett coolt ju!
A l l a   t e c k e n   h a r   s e p a r e r a t s .
S a e r s k r i v n i n g   p a a   n y   n i v a a !
```

TIPS: För att skriva ut något med färg eller en annan stil i terminalen anger vi en speciell teckenföljd. Alla tecken efter det kommer följa den stilen.

- `\x1b[1m` ger fetstilad text.
- `\x1b[32m` ger grön text.
- `\x1b[0m` nollställer stilen.
- Gles text får du genom att lägga in lämpliga blanksteg. Av tekniska skäl stöds endast ascii, dvs inte nationella tecken som åäö.

Exempel:

```
int main()
{
  std::cout << "\x1b[32m hej! \x1b[0m" << std::endl;
}
```

HJÄLP: Terminalkommandot `reset` brukar ofta kunna återställa en terminal som råkat få permanent konstig stil. Ett gångbart alternativ är att stänga terminalen och öppna en ny.

1337: Frivilligt. Tid över? För extra skryt implementerar du en extra klass som omvandlar strängen till leet-speak (O=0, L=1, R=2, E=3, A=4, S=5, G=6, T=7, B=8).

Uppgift 3 - Minneshantering - Påbyggnad för högre betyg

I filen `uppg3.cc` finns en implementation av en länkad struktur. Din uppgift är att komplettera koden så klassen `Stack` blir komplett, implementera de fem speciella medlemsfunktionerna och säkerställa korrekt minnehantering i alla lägen.

För att säkerställa att din lösning är korrekt ska du utöka det givna huvudprogrammet med följande testfall och kontrollera att inga minnesläckor uppstår:

1. Skapa en stack A med värden 2 5 8, kontrollera att A ej är tom och har 8 överst.
2. Skapa en stack B som en kopia av A, kontrollera att B ej är tom och har 8 överst.
3. Poppa B två gånger, kontrollera att B ej är tom och har 2 överst.
4. Pusha 4 till B, kontrollera att B ej är tom och har 4 överst.
5. Poppa A en gång, kontrollera att A ej är tom och har 5 överst.
6. Poppa A två gånger, kontrollera att A är tom.
7. Flytta B till A, kontrollera att B är tom, kontrollera att A ej är tom och har 4 överst.
8. Flytta A till en ny stack C, kontrollera att A är tom, kontrollera att C ej är tom och har 4 överst.
9. Skapa en stack D med värde 3, kontrollera att D ej är tom och har 3 överst.
10. Skriv över C med D, kontrollera att C ej är tom och har 3 överst.
11. Poppa C, kontrollera att C är tom, kontrollera att D ej är tom och har 3 överst.
12. Pusha 1 2 6 7 till D, avsluta programmet.

Uppgift 4 - Eko-värden - Påbyggnad för högre betyg

Skogsekologen Forest Stump har ansvar för naturvård och inventering av naturvärden i flera skogar och nationalparker. För att hålla reda på vad som är naturvärden har hen skapat ett index med ord som höjer ett träds värde i ekosystemet och gynnar andra arter. Väl ute i skogen antecknar Forest de naturvärda eller i framtiden potentiellt naturvärda träd hen ser. Nu vill hen ha din hjälp att programmera sitt `Eko_Index` så att det kan användas för att skapa `Forest_Report`s. Du ska skapa dessa båda klasser. Filen `uppg4.cc` innehåller ett huvudprogram som visar på hur klasserna ska kunna användas.

Klassen `Eko_Index` håller koll på en lista med ord. Varje ord symboliserar en egenskap som värderas ur ekologisk synpunkt. Eftersom Forest ska använda samma index till alla sina skogar ställer hen som krav att ett `Eko_Index` aldrig ska kunna kopieras. Funktionaliteten hos ett `Eko_Index` sammanfattas som:

- En konstruktor som tar in godtyckligt många ord.
- En funktion som givet ett ord svarar på om ordet finns i indexet eller ej.
- En funktion som givet en trädbeskrivning (flera ord separerade med mellanrum) räknar ut trädets totala eko-värde. Varje ord som förekommer i indexet höjer eko-värdet med 1. Till slut kvadreras summan för att ge det slutliga eko-värdet. (Kvadreringen gör att eko-värden som förekommer tillsammans är värda mer än summan av samma eko-värden som förekommer enskilt.) *Exempel*: Beskrivningen `"en stor gammal ek"` ger eko-poäng för `"gammal"` och `"ek"`, dvs summan 2. Summan kvadrerars enligt $2 * 2 = 4$. Slutliga eko-värdet är alltså 4.

Klassen `Forest_Report` räknar ut det samanlagda eko-värdet på en hel skog. För att göra detta behöver den tillgång till en instans av `Eko_Index` (utan att kopiera instansen!). Den håller även reda på skogens namn och en lista på alla trädbeskrivningar. Funktionaliteten sammanfattas som:

- En konstruktor som tar in nödvändiga parametrar.
- En funktion som lägger till en trädbeskrivning i rapporten.
- En funktion som summerar eko-värdet för alla trädbeskrivningar i rapporten.
- En funktion som returnerar skogens namn.

Körexempel (användarinmatning i fet stil)

```
$ ./a.out
OK
Mulleskogen: 23
Tiveden: 43
```

TIPS: Använd `std::initalizer_list` där så krävs.

KRAV: Utöver bedömningskraven på sida 1 får endast en instans av `Eko_Index` skapas. Din lösning ska vara korrekt uppdelad i deklarations (*.h) och implementations (*.cc) fil. Implementationen ska nyttja de funktioner som finns i den mån det går.