Procedurell Programmering
Inlämningsuppgift 1: Gissa ett tal

1. Uppgiftsbeskrivning

Som första inlämningsuppgift ska du utveckla ett C++ program där användaren ska utmana datorn genom att gissa på ett tal. Spelet går ut på att användaren gissar på ett tal som datorn slumpar fram. Gissar användaren rätt erhålls en vinst, annars förlust. Talet kan befinna sig inom ett intervall som användaren bestämmer innan spelet börjar. Beroende på intervallets storlek erhåller användaren ett antal gissningar.

I början av programmet ska användaren ange hur mycket pengar som ska satsas, 100kr, 300kr eller 500kr. Ju mer pengar användaren satsar, desto större intervall kan användaren ange. Om användaren gissar rätt tal, innan antal gissningar är slut, erhålls en vinst på dubbla insatsen. Vid förlust förloras insatsen.

Följande krav måste uppfyllas i ditt program:

- Vid start av programmet ska användaren kunna sätta in pengar som används inför varje spelomgång. Max 5000kr får sättas in.
- Finns inte några pengar innestående måste användaren sätta in mer pengar innan en spelomgång kan starta.
- Innan användaren väljer intervall ska pengarna satsas, 100kr, 300kr eller 500kr.
- När spelomgången börjar ska användaren mata in talets intervall. Beroende på insatsen kan användaren välja mellan följande intervall:
    - Intervallet 1-10 kräver insats på 100kr.
    - Intervallet 1 – 100 kräver insats på 100kr eller 300kr.
    - Intervallet 1 – 1000 kräver insats på 300kr eller 500kr.
- Tydligt framgå vilka intervall användaren kan välja beroende på aktuell insats. Ex, om användaren satsar 300kr ska intervallen 1-100 och 1-1000 synas. Eftersom 1-10 inte är tillåtet ska alternativet inte synas som val.
- Beroende på intervallets storlek ges följande antal gissningar innan spelet är slut:
    - 1 - 10 ger 3 gissningar.
    - 1 - 100 ger 5 gissningar.
    - 1 - 1000 ger 7 gissningar.
- Om användaren gissar rätt, ska spelomgången avslutas. Om användarens antal gissningar tar slut ska programmet också avslutas.
- Användaren får spela hur många spelomgångar som helst (baserat på att pengar finns innestående) och väljer själv när programmet ska avslutas.

- Lyckas användaren vinna ett spel ska programmet skriva ut aktuell vinst samt de totalt innestående pengarna.
- Om användaren inte vinner ett spel ska programmet skriva ut ett tröstmeddelande och den totala vinstsumma som finns kvar.
- Betalningsmomentet kan symboliseras med ex. information om att dess pengar dras från användarens konto. Använd gärna ett heltal som minskas och ökas beroende på aktuella vinster och förluster.
- Datorns tal kan symboliseras mha en fördefinierad randomfunktion, ex. rand(). Den returnerar ett slumpmässigt heltal. Tips: använd följande kod någonstans i ditt program:

    - `#include <ctime>; //Inkluderas i början av programmet`

    - `srand(time(0)); //Används en gång i början av programmet för att nollställa en timer.`

    - `rand() % 10 + 1; //Används för att slumpa fram ett värde mellan 1 och 10. Tilldela gärna det returnerade värdet till en variabel, ex int tal=rand()%10+1.`

- Det procedurella paradigmet måste användas för att lösa uppgiften. Objektorienterad programmering tillåts inte.

- Syntax för byggstenarna sekvens, iteration och selektion ska användas för att översätta flödesdiagram till källkod i programmeringsspråket C++ enligt kursmaterial. Hopp i programkoden som utförs med exempelvis ”goto”-kommandon är inte tillåtet.

För att kunna lösa ovanstående krav så måste du identifiera alla delproblem och därmed lösa dolda antaganden och krav som inte finns listade i uppgiftsbeskrivningen. I många fall så finns det stort utrymme för egen tolkning då har du möjlighet att lösa uppgiften på eget bästa sätt. Dock så är det viktigt att du tydligt beskriver vilka dolda antaganden och krav som du identifierat och löst i din dokumentation. Ett exempel på ett dolt antagande är att användaren får max sätta in 5000kr. Men om användaren förlorar ett spel tillåts det att han sätter in 5000kr igen.

2. Dokumentation

Din lösning ska lämnas in som en rapport och ska bestå av två delar, en laborationsrapport som beskriver lösningen och sedan bilagor i form av källkod. Lämpligt omfång på laborationsrapporten är 4 - 6 sidor (bilagor är inte inkluderade) då följande innehålla ska finnas:

- Försättsblad med namn, program och personnummer.
- Problembeskrivning där du beskriver vilka del problem som identifierats samt hur du har gått tillväga för att lösa dessa.
- Beskriv egna antaganden och krav som du identifierat för att lösa uppgifts listade krav.
- Flödesdiagram: en modell som löser uppgiftens problem.

Följande innehåll ska finnas med som bilagor:

- Källkod som löser uppgiften och som följer ditt flödesdiagram.
- Kommentarer i källkoden som tydligt förklarar avsikten med koden.

3. Regler

Inlämningsuppgiften ligger till grund för examination i kursen och löses individuellt. Detta innebär att det är väldigt viktigt att materialet som skickas in till läraren är ditt eget och är enkelt att identifiera som ditt eget. Att kopiera text, bilder eller kod är inte accepterat. Däremot är det tillåtet att använda källanvisningar för att visa vart information kommer ifrån samt diskutera lösningsförslag och strategier med lärare och studenter. Men uppgiften ska vara individuellt utförd efter egen förmåga. Grupparbete där källkod och skriftlig rapport skapas tillsammans tillåts inte.

4. Deadline

Sista inlämningsdatum för inlämningsuppgift 1 står angivet i Scio. Rapporten ska lämnas in elektroniskt via läroplattformen Scio (instruktioner finns på kurshemsidan). Alla andra former av inlämning (ex e-post, utskrivna papper) är inte tillåtna.